uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2018-2019

Trabajo Fin de Grado

"Diseño e implementación de módulos funcionales (Sensores y Telemetría) de OBSW para el desarrollo de nanosatélites"

Jaime García González

Tutor/es
Javier López Gómez
Leganés, 2019

"El coraje no es tener la fuerza para seguir; es seguir cuando no tienes fuerza."
Napoleón Bonaparte.

Resumen

Descubrir los misterios del Universo es el próximo desafío para la Humanidad. La filosofía *New Space* apuesta por la creación de satélites económicos que se construyan rápidamente para conseguir este objetivo. Este Trabajo de Fin de Grado forma parte de la Cátedra Universidad Carlos III - SENER, cuyo objetivo es el desarrollo de un nanosatélite que tome fotografías de la Tierra a alta resolución.

Los satélites son controlados por el software de a bordo, OBSW, que se ejecuta sobre un sistema operativo de tiempo real, RTOS. Como es un sistema crítico, las tareas se deben ejecutar correctamente y en los plazos establecidos. Si se incumple una de estas dos condiciones se pone en riesgo la misión, pudiendo ocasionar importantes pérdidas.

La monitorización del sistema es fundamental para el éxito de la misión, puesto que, se pueden corregir errores desde el segmento terrestre. Por ello, el objetivo de este Trabajo de Fin de Grado es el diseño e implementación del módulo de telemetría, así como, una simulación software de los componentes térmicos del sistema. Estos módulos se han realizado con el *framework* cFE de NASA, que facilita el desarrollo utilizando sus servicios.

Palabras clave: Nanosatélite, software de a bordo, sistema operativo de tiempo real, telemetría, simulación térmica.

Abstract

Discovering the mysteries of the Universe is the next challenge for Humanity. The New Space philosophy focuses on the creation of economic satellites that are built quickly to achieve this goal. This bachelor's thesis is part of the Cathedra Universidad Carlos III - SENER, whose objective is the development of a nanosatellite that takes high-resolution pictures of the Earth.

The satellites are controlled by the onboard software, OBSW, which runs on a real-time operating system, RTOS. Being a critical system, the tasks must be executed correctly and on schedule. If one of these two conditions is not met, the mission is put at risk and can cause significant losses.

Monitoring the system is fundamental for the success of the mission, because errors can be corrected from the ground segment. Therefore, the objective of this project is the design and implementation of the telemetry module, as well as a software simulation of the thermal components of the system. These modules have been made with the cFE framework of NASA, which facilitates the development using its services.

Keywords: Nanosatellite, on-board software, real-time operating system, telemetry, thermal simulation.

Agradecimientos

A mis padres, Cristina y Jaime, las personas más importantes de mi vida. Gracias por confiar siempre en mí, por vuestro apoyo cuando las cosas no salían bien, pero sobre todo, gracias por enseñarme que con esfuerzo y sacrificio no existe la palabra imposible.

A mis abuelas por su infinito amor y cariño, en especial a mi abuela Guada, que seguramente dentro de no mucho no recuerde nada de esto, te quiero abuela. A mi tío Esteban, por estar en los momentos más difíciles. A mis abuelos que hace muchos años que ya no están, sé que estarían orgullosos, siempre os llevaré en el corazón.

A Javier López, mi tutor de este proyecto, por su ayuda, compromiso y dedicación. Con seguridad una de las personas que más me ha marcado a nivel académico, un verdadero genio de la informática, gracias por transmitirme tu pasión.

A la Universidad Carlos III y a SENER por la oportunidad de participar en este proyecto. A Jesús Carretero y a Javier Fernández por su orientación y ayuda durante estos últimos meses.

Índice general

1. Introducción **1**

1.1. Motivación 1

1.2. Problemática 4

1.3. Objetivos 4

1.4. Estructura del documento 5

2. Estado del arte **7**

2.1. Exploración espacial 7

2.2. Proyectos académicos 11

2.3. Sistemas a bordo 12

2.4. Comparativa 17

3. Análisis y diseño **19**

3.1. Descripción del proyecto 19

3.2. Requisitos de usuario 25

3.3. Casos de uso 32

3.4. Requisitos de software 37

3.5. Diseño de la solución 45

3.6. Matrices de trazabilidad 52

4. Implementación **55**

4.1. Módulo de telemetría 55

4.2. Simulador térmico 61

4.3. Integración 63

5. Pruebas y evaluación **65**

5.1. Pruebas 65

5.2. Evaluación 70

6. Plan de proyecto **71**

6.1. Tareas 71

6.2. Planificación 72

6.3. Presupuesto 74

6.4. Entorno socio-económico 78

7. Marco regulador **81**

7.1. Legislación y normativa internacional 81

7.2. Legislación y normativa nacional 82

8. Conclusiones y trabajos futuros **83**

8.1. Objetivos 83

8.2. Proyecto 84

8.3. Personales 84

8.4. Trabajo futuro 84

A. Summary **87**

A.1. Introduction 87

A.2. cFE 89

A.3. System architecture 90

A.4. Implementation 91

A.5. Project plan 93

A.6. Conclusions 95

B. Glosario **97**

C. Manual de usuario **99**

Índice de figuras

1.1. Funcionamiento de los sistema de control 2
1.2. *Data link* entre OBSW y Ground System 3

2.1. Sputnik 1 [1] 8
2.2. Misión Apolo 11 [2] 9
2.3. Rover Opportunity [3] 10

3.1. Metodología de trabajo en V [4] 20
3.2. Arquitectura del sistema CFS 21
3.3. Arquitectura global del sistema 22
3.4. Plantilla de requisitos de usuario 25
3.5. Plantilla de casos de uso 32
3.6. Diagrama UML de casos de uso 33
3.7. Plantilla de requisitos software 37
3.8. Plantilla de componentes 45
3.9. Diagrama de componentes 46
3.10. Diagrama de secuencia de telemetría I (enviar paquete periódico) 50
3.11. Diagrama de secuencia de telemetría II (enviar paquete puntual) 50
3.12. Diagrama de secuencia control térmico I (dentro de rango) 51
3.13. Diagrama de secuencia control térmico II (fuera de rango) 51
3.14. Matriz de trazabilidad requisitos de capacidad-requisitos funcionales 52
3.15. Matriz de trazabilidad requisitos de restricción-requisitos no funcionales 53
3.16. Matriz de trazabilidad componentes - requisitos funcionales 53

4.1. Estructura de paquetes 55
4.2. Funcionamiento de Software Bus 56
4.3. Esquema del miembro *flags* 58

4.4. Función sinusoidal de temperatura 62

5.1. Plantilla de casos de prueba 66
5.2. Matriz de trazabilidad casos de prueba - requisitos funcionales 69

A.1. Data link between OBSW and Ground System 88
A.2. CFS system architecture 90
A.3. System architecture 91

C.1. Jerarquía de directorios del sistema 100
C.2. Jerarquía de directorios de un módulo del sistema 101

Índice de tablas

2.1. Comparativa entre OBSW 17

3.1. Características BeagleBoard 20

4.1. Bits del miembro *flags* 59
4.2. Tipos de paquetes 61

5.1. Tiempos de cómputo máximo y periodo 70
5.2. Intervalo de confianza 70

6.1. División de tareas 74
6.2. Costes de personal 75
6.3. Costes de material hardware 75
6.4. Costes de material software 75
6.5. Amortización de productos 76
6.6. Costes indirectos 76
6.7. Resumen de costes 77

A.1. Personnel costs 94
A.2. Amortisation costs 94
A.3. Indirect costs 94
A.4. Summary of costs 94

Capítulo 1

Introducción

Desde el inicio de los tiempos, la humanidad siempre ha deseado la capacidad de volar que poseen otros animales de la Tierra, otorgando este poder ¨sobrenatural¨ a muchos de sus dioses. Durante muchos siglos, se realizaron multitud de prototipos y máquinas que imitaban el vuelo de las aves.

Entre los siglos XV y XVI uno de los mayores genios de la historia, Leonardo Da Vinci, diseña multitud de artilugios [5] muy avanzados a su tiempo, entre ellos, un planeador que imitaba la forma de volar de las aves y un primitivo prototipo de helicóptero.

El 17 de diciembre de 1903, el sueño se hace por fin realidad, se consigue volar en una aeronave más pesada que el aire. Los hermanos Wright [6] consiguen recorrer con su biplano propulsado más de 250 metros en algo menos de un minuto.

En los siguientes años, el desarrollo de la aviación crece de manera exponencial, llegando a ser un pilar fundamental durante la II Guerra Mundial. Al terminar este conflicto, muchos ingenieros alemanes se incorporan a los programas espaciales de Estados Unidos y de la Unión Soviética dando comienzo a la carrera espacial. Finalmente, el 20 de julio de 1969 la misión Apolo 11 aterriza por primera vez en la superficie lunar.

Este desarrollo espectacular en un periodo de tiempo relativamente corto se debe a la colaboración de múltiples disciplinas como la aeronáutica, las telecomunicaciones, la informática, etc. Controlar el estado y gestionar el sistema es fundamental en este tipo de misiones, estas tareas son realizadas por los sistemas de control.

1.1. Motivación

Los sistemas de control se encargan de controlar el comportamiento de un entorno físico determinado. Estos sistemas realizan siempre el mismo algoritmo iterativamente. En primer lugar, se recibe información del entorno físico por medio de sensores. Seguidamente se analiza la información recibida y se decide que se debe modificar en el entorno físico para lograr el comportamiento deseado. Finalmente, si es necesario se realizan los cambios oportunos en el entorno físico utilizando actuadores, véase la Figura 1.1.

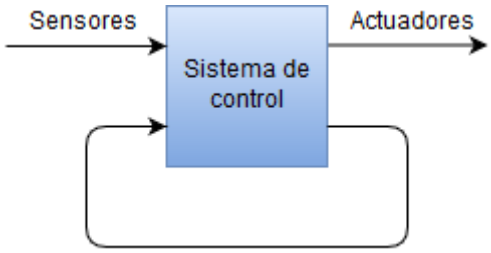


Figura 1.1: Funcionamiento de los sistema de control

El sistema de control a desarrollar se basa en un sistema en tiempo real, RTOS, crítico. Para que un sistema de control sea crítico debe cumplir dos condiciones. La primera condición es que el sistema debe ejecutar las acciones correctas. La segunda condición es que el sistema realice las acciones en el intervalo de tiempo establecido. El incumplimiento de una de estas dos condiciones implica el fallo total del sistema. En caso de que el sistema falle, las consecuencias son muy graves, tanto a nivel económico, como a nivel humano. Este tipo de sistemas son muy habituales en diferentes ámbitos tales como aviación, sistemas ferroviarios, control de automóviles, etc.

Los satélites artificiales utilizan RTOS, por tanto, siempre deben ejecutar las acciones correctas en los plazos de tiempo establecidos. En caso de no cumplirse ambas condiciones, el sistema falla totalmente, lo que implicaría el fracaso de la misión, que tendría como consecuencia la pérdida de cientos de millones de euros y de vidas humanas en caso de que el satélite cuente con tripulación. Por tanto, un buen diseño e implementación del sistema es esencial para conseguir el éxito de la misión.

El propósito de la cátedra conjunta entre la Universidad Carlos III y SENER, es desarrollar un nanosatélite, en concreto un *CubeSat* universitario, que funcione utilizando un RTOS. La misión de este *CubeSat* es realizar fotografías de alta resolución de la Tierra que se utilizarán con propósitos científicos. Al tratarse de un proyecto multidisciplinar, se requiere la participación de estudiantes de diferentes titulaciones de Grado, entre ellas el Grado en Ingeniería Informática.

El estándar *CubeSat* [7] se definió conjuntamente en el año 1999 entre la Universidad Politécnica de California y la Universidad de Standford, con el objetivo de desarrollar pequeñas naves espaciales en las aulas universitarias. Las características que debe cumplir un *CubeSat* son las siguientes:

Diseño. Las dimensiones y masa del sistema se definieron en el estándar de 1999. Las dimensiones del sistema son $10 \times 10 \times 10$ cm y una masa máxima de 1 kg. En años posteriores se decidió aumentar la masa máxima a 1,33 kg.

Modular. Se pueden desarrollar diferentes módulos funcionales (Módulo de comunicaciones, módulo de placas solares…) siguiendo el estándar para poder ser reutilizados en otros proyectos.

Lanzamiento. Normalmente estos nanosatélites se incluyen en cohetes que se vayan a poner en órbita para optimizar costes.

Presupuesto. Estos proyectos suelen tener presupuestos muy bajos, ya que, habitualmente se realizan con la única financiación de las universidades.

Riesgo. Estos proyectos suelen tener un nivel de riesgo muy alto, debido a su bajo presupuesto, rápida implementación y a la innovación y combinación de nuevos elementos.

La tarea asignada al Grado en Ingeniería Informática, consiste en el desarrollo e implementación del OBSW[1], es decir, el encargado del control y la gestión del nanosatélite en el segmento espacial. Además, se ha asignado el desarrollo e implementación del sistema del Ground System, que se encargará del control de la misión desde el segmento terrestre. Ambos segmentos tienen que estar en constante comunicación para comprobar que el sistema opera correctamente, por tanto, debe existir un enlace de comunicación (*data link*) permanente entre ambos, véase la Figura 1.2.

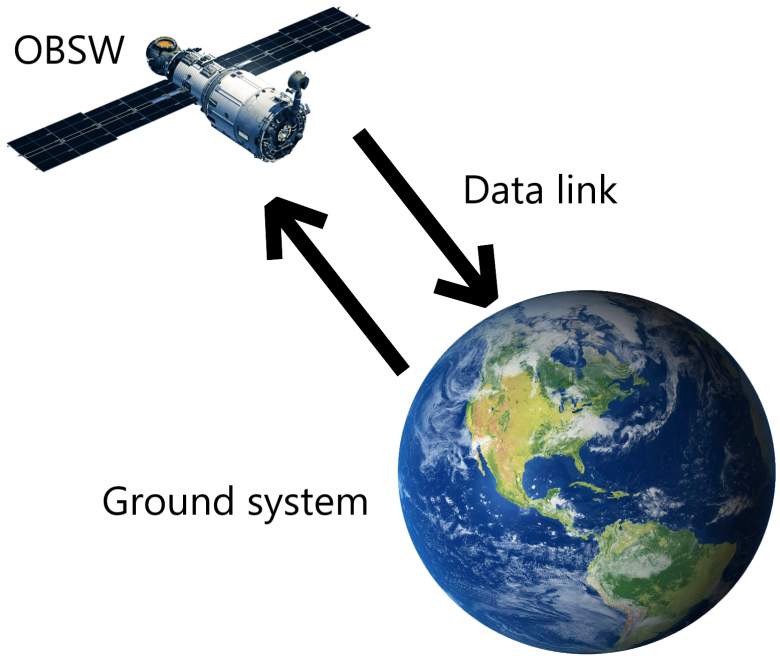


Figura 1.2: *Data link* entre OBSW y Ground System

El OBSW consta de diferentes módulos que consiguen que el sistema opere correctamente a nivel software y hardware. La tarea asignada en el presente Trabajo de Fin de Grado es el diseño y la implementación del módulo de telemetría y del simulador térmico, que se explicará en epígrafes posteriores.

[1]Véase la Sección 2.3

1.2. Problemática

El mal diseño e implementación de los diferentes módulos que componen el OBSW, implican el fallo total de la misión. Por tanto, se debe conocer en profundidad la arquitectura del sistema y cada uno de los componentes que lo forman. Asimismo, los componentes que forman el Ground System deben estar bien diseñados e implementados para observar el estado del nanosatélite desde el segmento terrestre y modificar su comportamiento si fuese necesario.

Todos los OBSW tienen una serie de módulos funcionales esenciales que permiten que el sistema se comporte correctamente. Los módulos iniciales que se desarrollarán en este proyecto son los siguientes:

Módulo de planificación. El *scheduler* o planificador, es el encargado de determinar el orden de ejecución de las tareas del sistema. Para ello, asigna a cada tarea una prioridad diferente y controla que se cumplen los tiempos de respuesta determinados.

Módulo de comunicaciones. El módulo de comunicaciones es el encargado de enviar y/o recibir datos del Ground System. Envía al segmento terrestre los datos obtenidos por los diferentes sensores del sistema y recibe telecomandos para ejecutar determinadas acciones. Para ello, se utilizarán *sockets* UDP.

Módulo de telemetría. El módulo de telemetría es el encargado de leer los datos que recogen todos los sensores que componen el sistema. Además, define la estructura de cada paquete de datos para cada uno de los sensores. Finalmente, debe reenviar los paquetes al OBSW utilizando el Software Bus[2]. Un mal comportamiento de este módulo, en lectura de datos del sensor o en envío de paquetes, supondrá el fallo total de la misión.

Módulo FDIR. El módulo FDIR (*Failure Detection Isolation and Recovery*) recibe los datos del sensor de telemetría y comprueba que todos los datos están dentro del rango preestablecido. Si los datos son correctos, se reenvían al módulo de comunicaciones, en caso contrario, se modificará el comportamiento del sistema para corregir errores.

1.3. Objetivos

A continuación, se detallan los objetivos que se pretenden conseguir con este proyecto:

Definición de la arquitectura del sistema. Se participará en el diseño de la arquitectura del sistema junto con los compañeros que trabajan en el resto de módulos de OBSW.

Diseño e implementación del simulador térmico. Definición, diseño e implementación de los diferentes sensores y actuadores térmicos que componen el sistema. Se simularán a nivel software estos componentes y el control térmico encargado encargado de su gestión.

Diseño e implementación del módulo de telemetría. Definición e implementación del módulo de telemetría. Se deben definir e implementar los paquetes. Además, se debe conseguir comunicación con los módulos del OBSW mediante el Software Bus.

[2]Explicación del Software Bus en la Sección 4.1

1.4. Estructura del documento

En la presente sección se detalla el contenido de los capítulos que conforman este documento. El documento tiene la siguiente estructura:

Estado del arte. Este capítulo realiza un estudio sobre los principales hitos de la exploración espacial, además, se incluye información relevante misiones espaciales académicas similares. Finalmente, se realiza un estudio sobre los principales software de a bordo (OBSW) y se hace una comparativa de los mismos.

Análisis y diseño. Este capítulo incluye los requisitos de usuario, casos de uso, así como, los requisitos del sistema derivados de los requisitos de usuario. Además, se detalla la arquitectura global del proyecto y del presente Trabajo de Fin de Grado. Finalmente, se incluyen las matrices de trazabilidad entre los requisitos y componentes del sistema.

Implementación. Este capítulo incluye los detalles más relevantes de la implementación de los módulos del sistema.

Pruebas y evaluación. Este capítulo incluye las pruebas que se han realizado para verificar el correcto comportamiento del sistema. Además, se evalúan los tiempos de cómputo de cada módulo y se verifica que se cumplen las restricciones de tiempo.

Plan de proyecto. Este capítulo incluye la división de tareas del proyecto y el diagrama Gantt donde se detalla su planificación temporal. Además, incluye el presupuesto y el impacto socio-económico del mismo.

Marco regulador. Este capítulo incluye la legislación y normativa actual que debe cumplir el proyecto.

Conclusiones. Este capítulo incluye las conclusiones obtenidas después de realizar el proyecto, además de las tareas que se podrían realizar en el futuro.

Capítulo 2

Estado del arte

En el presente capítulo se analizan los primeros pasos de la humanidad en la exploración espacial, además de diferentes alternativas de OBSW, así como, diversas misiones espaciales realizadas con propósitos educativos. En primer lugar, en la Sección 2.1, se explicarán los primeros avances científicos que contribuyeron al estudio de los astros, la carrera espacial entre Estados Unidos y la Unión Soviética y finalmente se comentarán algunas de las misiones espaciales más importantes hasta la fecha. Seguidamente, en la Sección 2.2, se analizarán diferentes misiones realizadas por agencias espaciales o compañías privadas en colaboración con universidades para poner en órbita nanosatélites con propósitos educativos. Finalmente, en la Sección 2.3, se analizarán diferentes herramientas de control que son utilizadas en misiones reales, tanto en el control terrestre como en el segmento espacial.

2.1. Exploración espacial

Primeros pasos

La exploración espacial siempre ha sido un tema de interés para la humanidad. Las primeras civilizaciones conocidas como la azteca, la mesopotámica y la hindú, construyeron grandes templos en honor a sus dioses que provenían de las estrellas. La civilización griega y la árabe fueron capaces de medir y calcular las órbitas de los astros, de esta forma, se elaboraron calendarios muy precisos. Todas estas mediciones se realizaban sin instrumentación, por ello, tenían cierto margen de error.

La invención del telescopio, supuso un gran avance y revolución en este campo. La autoría de la invención del telescopio es muy confusa, siempre se pensó que fue el alemán Hans Lippershey su creador, sin embargo, estudios recientes han demostrado que el invento original fue de Joan Roget, un fabricante de lentes francés que vivía en Cataluña.

Aprovechando la patente del telescopio, en 1609, Galileo Galilei [8], un astrónomo italiano, presentó su propio telescopio que utilizó para realizar múltiples observaciones astronómicas. Una de sus aportaciones principales fue el modelo heliocéntrico, que proponía que la Tierra giraba alrededor del Sol, frente al modelo geocéntrico clásico. Esto provocó la ira de la Iglesia católica que censuró sus trabajos y le condenó a prisión perpetua, prohibiéndole salir de su casa en Florencia.

Los primeros cohetes se comenzaron a construir a finales del siglo XIX y principios del siglo XX, únicamente con propósitos científicos. Sin embargo, en pleno conflicto bélico, durante la II Guerra Mundial, los ingenieros alemanes presentan el cohete V-2, el primer misil balístico capaz de realizar vuelos suborbitales y que fue utilizado para el bombardeo de múltiples ciudades europeas. Es considerado como el padre de los cohetes modernos y sentó las bases para la carrera espacial.

Carrera espacial

En 1945, Alemania pierde el conflicto frente al bloque de los Aliados. En ese momento, Estados Unidos y la Unión Soviética, recopilan toda la información posible del programa espacial alemán, llegando a incorporar a algunos de sus ingenieros a su propio programa espacial.

En los años posteriores da comienzo la Guerra Fría entre dos grandes bloques debido a las importantes diferencias ideológicas. Por un lado, se encuentra el bloque capitalista liderado por Estados Unidos y por el otro el bloque comunista liderado por la Unión Soviética.

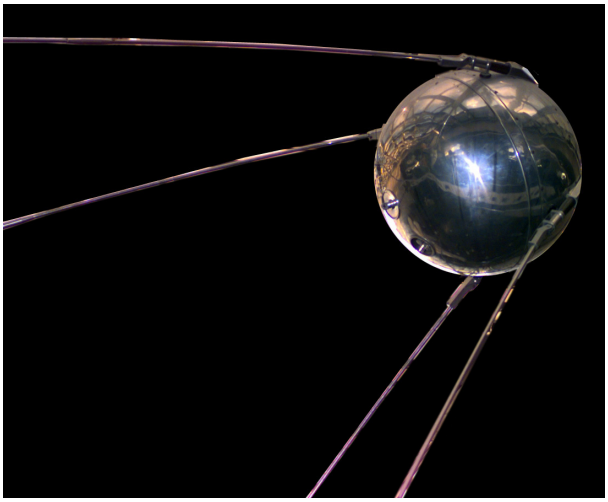


Figura 2.1: Sputnik 1 [1]

El 4 de octubre de 1957 se produce un hecho histórico en la carrera espacial, la URSS pone en órbita el primer satélite artificial realizado por la humanidad, el Sputnik 1 [9].

Un mes más tarde, el programa espacial soviético pone en órbita el Sputnik 2, el primer satélite que lleva a bordo un ser vivo, la perra Laika. La misión no contemplaba el retorno de la nave, por ello, incluyeron comida envenenada que suministrarían al animal a partir del décimo día para evitar su sufrimiento, sin embargo, en 2002 se demostró que Laika murió por un sobrecalentamiento del satélite.

Amenazados por el desarrollo tecnológico soviético, Estados Unidos lanza su primer satélite, el Explorer I, el 31 de enero de 1958. Durante los años siguientes el número de satélites puestos en órbita por ambos bloques creció exponencialmente. La mayoría de estos satélites se utilizaban para las comunicaciones, el estudio de la meteorología y para espionaje.

El 12 de abril de 1961, el soviético Yuri Gagarin fue el primer hombre en realizar un vuelo orbital, que duró únicamente 48 minutos. Dos años más tarde, el 19 de junio de 1963, la URSS

puso en órbita la Vostok 6 y Valentina Tereshkova se convirtió en la primera mujer que llegó al espacio.

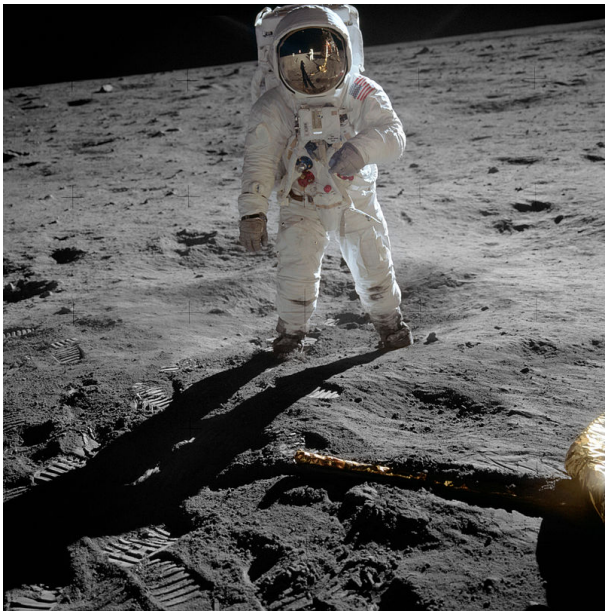


Figura 2.2: Misión Apolo 11 [2]

El 16 de julio de 1969 da comienzo la misión Apolo 11. El 20 de julio de 1969 se produce uno de los eventos más importantes del siglo XX y de la historia de la humanidad, Neil Armstrong, Buzz Aldrin y Michael Collins llegan a la Luna. Unas horas después de haber alunizado, Neil Armstrong se convierte en el primer hombre en pisar la superficie lunar y pronuncia la siguiente frase:

"It's one small step for man, one giant leap for mankind ".

Cuatro días más tarde, el 24 de julio, la misión se completó satisfactoriamente al aterrizar el módulo lunar en el océano Pacífico. El momento del paseo lunar fue retransmitido a todo el planeta y tuvo un grandísimo impacto social.

Exploración de Marte

Marte es el planeta más próximo a la Tierra, por ello, se han realizado múltiples misiones espaciales [10] para conocerlo mejor. Se han realizado más de 40 misiones a Marte, más de la mitad de ellas fracasaron.

La primera misión que tuvo éxito, fue la Mariner 4, lanzada por la NASA en 1964. Su objetivo era pasar cerca del planeta rojo y tomar fotografías, la misión se completó satisfactoriamente.

La primera nave que fue capaz de entrar en la órbita de Marte fue la Mars 2, de la Unión Soviética en 1971. Su versión mejorada, la Mars 3, entró en órbita y consiguió soltar el módulo de aterrizaje, sin embargo, fue destruido por una tormenta de arena.

Sin embargo, fue el programa Viking de la NASA, formado por las naves Viking I y Viking II, el que más información aporto de Marte hasta la década de los 90. Ambas naves consiguieron aterrizar correctamente y realizaron las primeras fotografías detalladas de la superficie marciana.

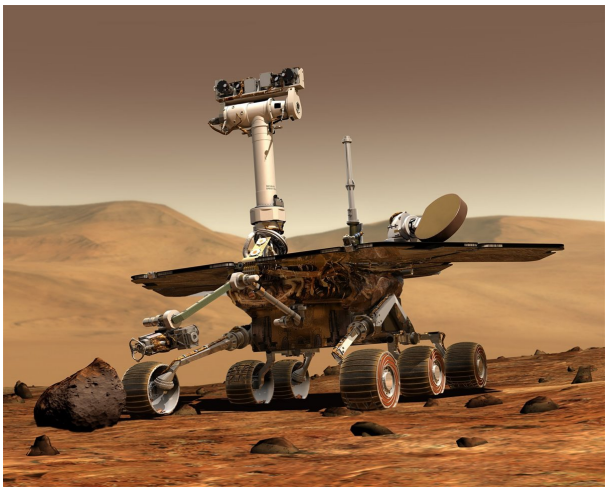


Figura 2.3: Rover Opportunity [3]

El 25 de enero de 2004 el Opportunity [11] es el segundo vehículo robótico, rover, de la NASA que aterriza en Marte como parte de su programa de exploración.

Se estimó que la misión duraría unos 90 días marcianos, aunque muchos miembros del proyecto creían que el robot podría funcionar más tiempo. Una vez que se superó el plazo estimado, la NASA aportó más fondos para continuar con la misión.

Algunos de los principales logros que consiguió el Opportunity son los siguientes:

- Encontrar evidencias de agua.
- Encontrar jarosita, una roca que se forma en medios expuestos al agua.
- Varios años de investigación en el cráter Victoria.
- Perfil de temperatura atmosférica.

Durante los 15 años que estuvo en activo, recorrió más de 45 kilómetros, convirtiéndose en el vehículo de exploración que más distancia ha recorrido hasta la fecha. En todo este tiempo, el Opportunity ha sido capaz de escalar cráteres y soportar muchas tormentas de arena, consiguiendo más de 5000 días marcianos en la superficie.

El 10 de junio de 2018 el vehículo de exploración entró en modo de hibernación debido a una fuerte tormenta de arena. Sus paneles solares quedaron cubiertos de polvo, por tanto, no se podían recargar sus baterías. La NASA mantuvo la esperanza de volver a establecer comunicación con el Opportunity, pero, después de 835 intentos sin respuesta, el 13 de febrero de 2019 finalizó la misión.

Realizar viajes tripulados a Marte supondría grandes desafíos para la humanidad a corto y medio plazo, algunos de ellos son:

- Proteger a la tripulación de la radiación.
- Generar energía para el trayecto y generar oxígeno y agua para los humanos.
- Prepararse mentalmente para el aislamiento.

Esto supone un gran reto a corto plazo, por ello, las principales agencias espaciales consideran primordial volver a la Luna y establecer bases allí. En concreto, la ESA y la NASA trabajan conjuntamente para volver al satélite terrestre en la próxima década.

2.2. Proyectos académicos

SiriusSat

Los satélites SiriusSat [12] son nanosatélites educacionales que han sido construidos por estudiantes en colaboración con la compañía privada *Sputnix*, que es experta en este ámbito.

Los nanosatélites desarrollados siguen el estándar *CubeSat* que se propuso en el año 1999 por la Universidad Politécnica de California y por la Universidad de Standford, con el objetivo de llevar proyectos espaciales a las universidades. Tiene como objetivo permitir a los estudiantes participar en el proceso de diseño y construcción de nanosatélites con unas prestaciones similares al del *Sputnik*, el primer satélite puesto en órbita por la humanidad.

El SiriusSat-1 tiene un peso de 1.45 kg y tiene como objetivo el estudio de la meteorología en el espacio, por ello, incorpora un detector de partículas desarrollado por por el Instituto de Física Nuclear de Moscú. Estos satélites tienen las certificaciones correspondientes y fueron puestos en órbita en el año 2018. Su sistema se encarga de transmitir balizas (*beacons*) cada 30 segundos al segmento de tierra.

Adicionalmente, los radioaficionados pueden participar en el proceso de recepción de señales. Estos satélites incorporan un software de decodificación de telemetrías, SPUTNIX Telemetry Viewer, que se encarga de proveer datos en texto y en gráficos para monitorizar y comprender la información recibida desde el segmento espacial.

Fly Your Satellite!

Fly Your Satellite! [13] es un programa propuesto por la ESA para que los estudiantes universitarios europeos puedan poner en órbita su propio *CubeSat*. Todos los equipos que forman parte de este programa, son supervisados y guiados por empleados de la compañía.

En la primera edición del programa participaron siete grupos universitarios. Sus contribuciones fueron las siguientes:

e-st@r Los alumnos de la universidad politécnica de Turín se encargaron del desarrollo de un sistema de determinación y control de altitud.

Goliat Los alumnos de Rumanía se encargaron de desarrollar un sistema para tomar fotografías de la superficie terrestre y para medir la radiación.

Masat-1 Este *CubeSat* fue desarrollado por alumnos húngaros, que se encargaron de implementar un sistema de control de energía y del manejo de datos.

PW-Sat Desarrollado por alumnos polacos, este sistema se encarga de comprobar que el sistema de reducción de aire, *drag*, funciona correctamente y es capaz de sacar de órbita al nanosatélite.

Robusta Desarrollado por alumnos franceses de la universidad de Montpellier, este *CubeSat* se encarga de testear y evaluar los efectos de la radiación en los transistores de los componentes electrónicos.

UniCubeSat-GG Este *CubeSat* fue desarrollado por los alumnos de la universidad de Roma para estudiar los efectos de la excentricidad orbital en el movimiento de altitud.

XaTcobeo Desarrollado por estudiantes españoles de la universidad de Vigo, este *CubeSat* se encarga de realizar comunicaciones por radio y desplegar paneles solares.

Estos siete nanosatélites, se incluyeron como carga del cohete Vega [14]. Este cohete fue desarrollado por la Agencia Espacial Italiana y por la ESA. Su viaje inaugural se realizó el 13 de febrero del año 2012. Se estimó que los *CubeSat* tendrían una vida orbital de unos cuatro años, que se podría reducir en función de la resistencia atmosférica.

UPSat

UPSat [15] es el primer *CubeSat* construido por la Libre Space Foundation en colaboración con la Universidad de Patras.

Su objetivo principal era ser el primer satélite *open source*, a nivel software y hardware, puesto en órbita. La primera tarea que realizó fue transmitir correctamente datos científicos desde el módulo de comunicación al segmento terrestre. Su segunda tarea fue realizar fotografías de la Tierra y en especial de Grecia desde el espacio y transmitirlas a tierra.

Base de datos de nanosatélites

Si se desea consultar más información sobre misiones espaciales de nanosatélites, se recomienda acceder a Nanosats Database [16], que contiene un listado con todas las misiones registradas hasta la fecha.

2.3. Sistemas a bordo

El OBSW [17] es el software que se ejecuta en el satélite durante la misión. Se ejecuta de forma aislada e independiente, controlando todo el comportamiento de la plataforma, desde la propulsión de los motores hasta los sensores de datos. En la mayoría de misiones, el OBSW es capaz de recibir, interpretar y ejecutar telecomandos que recibe del Ground System por un *data link*, todo ello en tiempo real.

Estos sistemas empotrados además de cumplir las restricciones habituales (bajo coste, pocos recursos, ejecución en tiempo real…) tienen que estar preparados para soportar las duras condiciones espaciales. Algunas de las limitaciones espaciales son las siguientes:

- Radiación espacial.
- En caso de fallo, no es posible reparar el sistema manualmente.
- Restricciones de dimensiones y masa para el lanzamiento.

Simulus

Los simuladores de satélites del *Centro Europeo de Operaciones Espaciales* (ESOC) están construidos bajo la infraestructura de Simulus [18]. Esta herramienta, está formada por un *framework* en tiempo real, diversos emuladores software, un conjunto de modelos genéricos que pueden ser reutilizados y por un conjunto de modelos del segmento de tierra.

Los desarrollos más recientes, se fundamentan en una arquitectura referencial, que permite identificar, definir, desarrollar e integrar una referencia operacional en la arquitectura del simulador. Además, este tipo de arquitectura permite reducir costes debido a su alto grado de reusabilidad.

El elemento principal de esta herramienta es su emulador de OBSW, que se utiliza para simular una misión real en la fase de desarrollo mientras las piezas reales del satélite se modelan. Esto permite potenciar el realismo de la simulación, además de verificar y validar el comportamiento del OBSW.

Simulus proporciona un conjunto de instrucciones software para diferentes microprocesadores que han sido aprobados para diversas misiones espaciales (SPARC V7, ERC32…).

Todos los modelos de simulación, incluyendo el emulador, se ejecutan bajo el entorno SIMSAT. SIMSAT está formado por diferentes módulos que permiten al kernel simulaciones en tiempo real. El kernel incluye un planificador y un motor de ejecución que permite el manejo de eventos.

Sentinel-3 Toolbox

El Sentinel-3 Toolbox [19] es un paquete formado por diferentes herramientas de visualización, procesamiento y análisis utilizados en la misión Sentinel-3. Esta misión forma parte del proyecto Copérnico de la ESA.

El proyecto Copérnico [20] está formado por una constelación de satélites que describen órbitas bajas terrestres (LEO) y cuyo objetivo es tomar fotografías de alta resolución de la Tierra para que estén disponibles para la comunidad científica o para cualquier persona interesada. El primer Sentinel-3A fue lanzado en 2016, mientras que, el Sentinel-3B fue lanzado en 2018. Las últimas misiones Sentinel están planificadas para el año 2021.

Las características principales [21] del conjunto de herramientas Sentinel-3 Toolbox son las siguientes:

- Rápida visualización de imágenes pesadas.
- *Framework* de procesamiento gráfico.
- Acceso a las bases de datos.

- Herramienta de extracción de píxeles.

KubOS

KubOS [22] es un *framework* para satélites de código libre y está diseñada para aumentar la productividad en el desarrollo, reducir el riesgo y permitir que los equipos se centren en el *payload*.

KubOS está empaquetado en Linux, cada subsistema tiene su propia API e incorpora servicios de núcleo. De esta forma, permite que los desarrolladores se centren en implementar nuevas aplicaciones para la misión de una forma más sencilla. El sistema sigue una arquitectura modular dividida en capas, es la siguiente:

1. Aplicaciones de la misión.
2. Servicios de núcleo.
3. Subsistema de API.
4. Kernel Linux.
5. Hardware del satélite.

Esta compañía ha desarrollado Major Tom [23], un sistema de control para el Ground System. Major Tom permite al usuario realizar un control de misión desde una interfaz muy intuitiva. Además, permite el control de varias misiones simultáneamente.

Algunas de las funciones principales que ofrece este sistema, son las siguientes:

- Envío de telecomandos al OBSW.
- Transferencia de archivos que permite actualizar el sistema e incorporar nuevas funciones.
- Control de telemetría.
- Sistema de notificaciones y alertas basado en eventos.
- Seguimiento de múltiples satélites, además de un predictor de órbita.
- Integración con KubOS.

CubedOS

CubedOS [24] fue desarrollado con el objetivo de proporcionar un software robusto para las misiones *CubeSat* y para facilitar el desarrollo de aplicaciones. CubedOS es muy similar al OBSW desarrollado por NASA, cFE. Se diferencian en que CubedOS está escrito en SPARK para evitar que existan errores en tiempo de ejecución.

El sistema está dividido en diferentes capas, logrando una abstracción del hardware. La mayor parte de los módulos del sistema están escritos en SPARK para evitar errores durante la ejecución, sin embargo, también se pueden programar módulos en Ada o en C.

Las principales características que ofrece este OBSW, son las siguientes:

- Un sistema asíncrono de paso de mensajes. Este sistema y el entorno de ejecución de Ada, constituyen el *kernel* de CubedOS.
- Librerías en tiempo de ejecución verificadas con SPARK.
- Módulo de reloj en tiempo real.
- Interfaz para el sistema de ficheros.
- Interfaz para comunicación radio.
- Diferentes módulos que ofrecen soporte para el protocolo CCSDS.

CubedOS necesita diferentes *drivers* para lograr que todos los componentes hardware operen correctamente. Los *drivers* varían entre misiones, por tanto, CubedOS utiliza un modelo de *driver* genérico que permite que los componentes se comuniquen entre ellos.

Las principales ventajas de este sistema, son las siguientes:

- La arquitectura de paso de mensajes es concurrente y permite que las tareas se solapen.
- Los programas se pueden adaptar al entorno de ejecución y comunicación.
- La arquitectura es consistente con las restricciones de Ada Ravenscar.

Core Flight Executive

Core Flight Executive [25], cFE, es una aplicación de desarrollo y un entorno de ejecución desarrollado por la Administración Nacional de la Aeronáutica y del Espacio de los Estados Unidos, NASA.

Este entorno proporciona un conjunto de servicios que facilitan el desarrollo e implementación de módulos para misiones espaciales. Los servicios que incorpora son los siguientes:

Software Bus Este servicio es el encargado de establecer la comunicación entre todos los componentes del sistema.

Time Este servicio implementa un sistema de control para controlar el inicio y fin de las tareas y notificarlo al segmento terrestre.

Events Este servicio es muy potente, ya que, combinado con el servicio de tiempo permite tener un registro total de todas las acciones que han sido ejecutadas por el sistema.

Executive Este servicio se encarga del inicio y ejecución del entorno.

Table services Este servicio permite definir las tareas que debe ejecutar el sistema, su prioridad, etc. Es decir, es el planificador del sistema.

El entorno cFE proporciona a los desarrolladores la interfaz de programación de la aplicación, API, de cada uno de los servicios que implementa. Además, incluye una serie de herramientas de desarrollo que son esenciales en estas misiones, son las siguientes:

- *Framework* de test unitarios que permiten a los usuarios comprobar que las aplicaciones desarrolladas funcionan correctamente.
- Analizador de tiempo que proporciona el rendimiento en tiempo real. Esta herramienta es esencial, ya que, permite conocer el comportamiento del sistema en un sistema empotrado antes de extrapolarlo al hardware real de la misión.
- Constructor de tablas de tareas para el planificador.
- Utilidades para gestión de comandos y telemetría.

Sin embargo, cFE es uno de los componentes que forman CFS, es decir, el núcleo del sistema de vuelo. CFS es un proyecto software independiente que se permite reutilizar, así como, sus aplicaciones.

La misión principal de cFE es asentar las bases de una plataforma que utilice software independiente reusable. Además, junto con la capa de abstracción del sistema, OSAL, se consigue que el código funcione en cualquier sistema operativo en tiempo real (VxWorks, RTEMS…) sin tener que modificarse. También, se persigue crear un estándar para estas misiones, evitando que cada agencia espacial utilice el suyo propio.

En conclusión, el propósito de cFE es desarrollar software para sistemas empotrados que puedan ser probados en ordenadores personales sin necesidad de modificar el código original.

Algunas de las mejoras que contempla el equipo de desarrollo que se podrían implementar en el futuro para facilitar el desarrollo de módulos y aplicaciones para este entorno son las siguientes:

- Añadir una interfaz gráfica, GUI, que permita simular el envío de comandos y la recepción de telemetría.
- Mejoras en comunicaciones distribuidas.
- Integrar un IDE para facilitar el desarrollo de aplicaciones.
- Añadir un modelo de memoria protegido.

El sistema utiliza una arquitectura [1] modular segmentada en capas.

NOS3

NOS[3] [26] es una herramienta que permite a los desarrolladores construir y probar el software de vuelo con modelos hardware simulados. El sistema no tiene conocimiento de que se está ejecutando en la Tierra y actúa como si se estuviese ejecutando en el espacio. Por ello, obtiene los datos de los sensores normalmente.

Este simulador incluye un conjunto de herramientas para realizar simulaciones de *CubeSat* en máquinas virtuales. Algunos de los componentes que integra son los siguientes:

[1] En la Sección 3.1 se detalla la arquitectura del sistema.

NOS. Es el núcleo de NOS^3, desarrollado por JSTAR para simular los buses de comunicación a nivel hardware y software.

CFS. Es un *framework* de código abierto, independiente y reutilizable desarrollado por NASA.

Simuladores hardware. Emuladores de los diferentes componentes hardware que forman la misión. Se pueden personalizar según las necesidades del usuario.

Vagrant. Permite al computador configurar la máquina virtual para ejecutar las aplicaciones que incorpora NOS^3.

COSMOS. Proyecto de código abierto que se utiliza para enviar comandos y controlar el OBSW.

OIPP. Es una herramienta desarrollada por NASA que permite al Ground System conocer la posición del satélite, así como, su posición respecto al Sol.

42. Es una herramienta de simulación y visualización desarrollada por NASA que permite al usuario conocer la órbita que describe el satélite. Recupera los valores del campo magnético y de posición del magnetómetro y del GPS.

NOS^3 se combina con cFE para proporcionar un entorno de simulación totalmente funcional sin la necesidad de utilizar hardware.

2.4. Comparativa

En la Tabla 2.1, se muestra una comparativa entre los OBSW expuestos en la Sección 2.3.

Característica	Simulus	KubOS	CubedOS	cFE + NOS^3
Simulación de hardware	✓	✓	✗	✓
Pruebas unitarias	✗	✗	✗	✓
Telemetría y telecomandos	✓	✓	✓	✓
Desarrollado en	C++	C, Rust y Python	SPARK, Ada y C	C
Open Source	✓	✓	✓	✓
RTOS	✗	✗	✗	✓
Entorno de escritorio	✓	✓	✗	✓

Tabla 2.1: Comparativa entre OBSW

Los OBSW expuestos anteriormente son bastante similares, todos permiten al desarrollador crear nueva funcionalidad de manera sencilla y sin tener que preocuparse de los componentes, es decir, abstrayéndose de la capa hardware.

CubedOS ofrece la misma funcionalidad que cFE, ya que, se desarrolla a partir de su código fuente, con la diferencia de que está desarrollado en SPARK para corregir errores en tiempo de ejecución.

Simulus y KubOS ofrecen la misma funcionalidad, se diferencian únicamente en el lenguaje en el que ha sido desarrollado. Carecen de un sistema para realizar pruebas unitarias y comprobar el correcto funcionamiento de todos los componentes desarrollados.

A priori, se puede pensar que cFE es el OBSW más incompleto, sin embargo, si se combina con NOS^3 se obtiene el mejor entorno de desarrollo y de simulación.

Las principales razones que han sido decisivas para elegir cFE como OBSW para la misión son las siguientes:

1. Requisito del cliente.
2. Soporte para RTOS, en particular soporta RTEMS.
3. Desarrollado por NASA y utilizado en diversas misiones reales.
4. Servicios que ofrece el sistema (Software Bus, Events, Time…).
5. Extensa y documentada API.

Capítulo 3

Análisis y diseño

En el presente capítulo se expone el proceso de análisis y de diseño del sistema. En primer lugar, en la Sección 3.1 se describe el proyecto. Seguidamente, en la Sección 3.2 se realiza el proceso de obtención de requisitos de usuario. En la Sección 3.3 se incluye un diagrama UML con los casos de uso del sistema. Seguidamente, en la Sección 3.4 se incluyen los requisitos software derivados de los requisitos de usuario. En la Sección 3.5 se realiza el diseño de la solución. Finalmente, en la Sección 3.6 se incluyen las matrices de trazabilidad del sistema.

3.1. Descripción del proyecto

El objetivo de la Cátedra Universidad Carlos III - SENER es desarrollar un *Cubesat*, es decir, un modelo de nanosatélite. Este nanosatélite debe tomar fotografías de la Tierra a alta resolución que serán utilizadas con propósitos científicos y académicos. Para ello, el nanosatélite describe una órbita terrestre baja (LEO) [27].

El objetivo de este Trabajo de Fin de Grado es el diseño e implementación de diferentes sensores del sistema, así como, del módulo de telemetría del OBSW. Para ello, se ha seguido el estándar PUS [28] de la ESA, en concreto los siguientes:

Housekeeping (PUS 3). Definición de paquetes de telemetría.

Real Time Forwarding Control (PUS 14). Control de envío de paquetes de telemetría.

On-board storage and retrieval (PUS 15). Almacenamiento y recuperación de paquetes de telemetría.

Al ser un proyecto transversal que requiere colaboración interdisciplinar, los requisitos y objetivos de la misión se modifican frecuentemente utilizando metodología ágil. Por tanto, el método de trabajo elegido es la "metodología en V" [29], véase la Figura 3.1. Esta metodología se caracteriza por ser rápida y barata, consta de cuatro niveles.

Análisis de requisitos. Se obtienen los requisitos del producto según las necesidades del cliente.

Análisis funcional. Se define la funcionalidad que debe cumplir el sistema, se realiza en base a los requisitos del cliente.

Arquitectura del sistema. Se definen los componentes hardware y software del sistema.

Implementación. Esta fase consiste en el desarrollo de la funcionalidad necesaria para que opere correctamente el sistema.

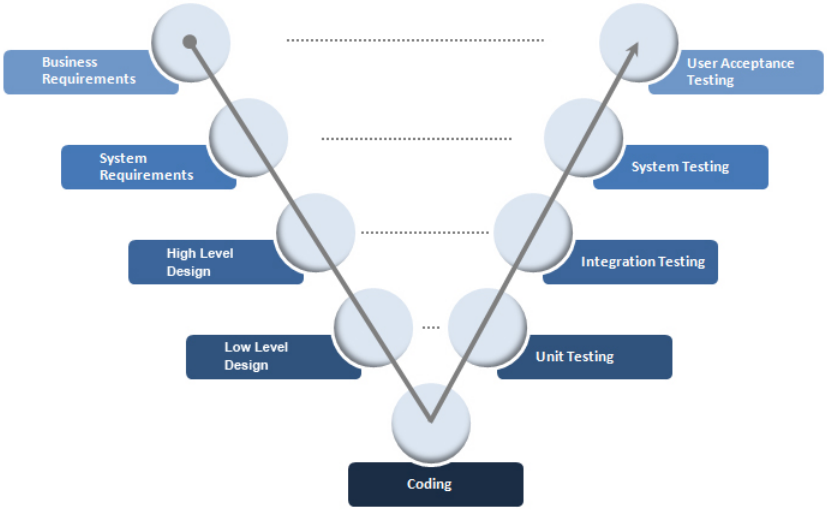


Figura 3.1: Metodología de trabajo en V [4]

Entorno operacional

El software ha sido desarrollado para ser ejecutado en dispositivos con muy pocos recursos. El entorno operacional sobre el que debe ejecutarse el sistema es una placa BeagleBoard-xM [30], cuyas características principales se encuentran en la Tabla 3.1.

Placa	Procesador	RAM	M.externa	USB	Conexión	RTEMS
BeagleBoard-xM	ARM Cortex-A8	512MB	MMC/SD	4 x USB 2.0	4 x Ethernet	✓

Tabla 3.1: Características BeagleBoard

Visión general

El sistema CFS sigue una arquitectura modular segmentada en capas, véase la Figura 3.2. Las capas que componen este sistema son las siguientes:

Hardware. Son los diferentes componentes hardware que forman el sistema.

Operating System. Sistema operativo sobre el que se ejecuta el sistema. Puede ser un RTOS (RTEMS, VxWorks…) o no (Linux).

OSAL. Capa que abstrae al sistema del OS o RTOS sobre el que se ejecuta la misión.

cFE Services. Servicios que ofrece el sistema cFE, explicados en la Sección 2.3.

CFS/ User Applications/Libraries. Conjunto de aplicaciones de ejemplo que incluye el sistema y aplicaciones desarrolladas por el usuario.

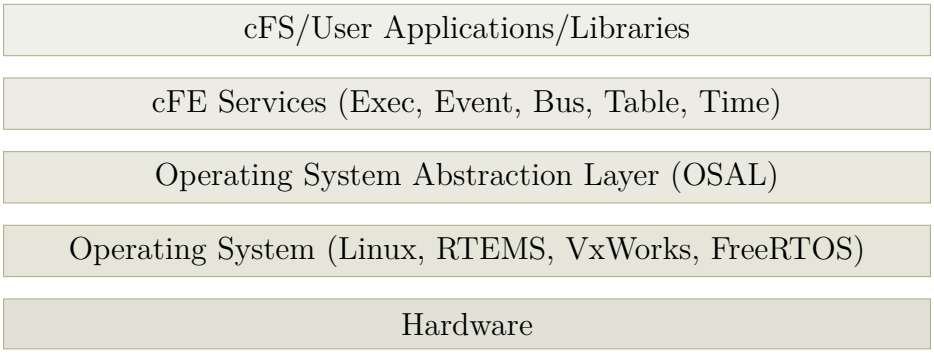


Figura 3.2: Arquitectura del sistema CFS

La arquitectura del sistema para esta misión ha sido definida por los integrantes del OBSW y del Ground System conjuntamente con el jefe de proyecto. Esta arquitectura se corresponde con la primera fase de diseño del proyecto y se modificará según avance la Cátedra hasta que se ponga en órbita el nanosatélite.

El sistema consta de tres subsistemas diferenciados, son los siguientes:

Hardware del sistema. Componentes hardware (sensores y actuadores) que se encargan de comprobar el entorno y actuar en consecuencia. En esta primera fase de desarrollo, el hardware será simulado, ya que, los componentes se corresponden con diferentes Trabajos de Fin de Grado de otros departamentos.

OBSW. El OBSW es el software de a bordo de la misión, consta de diferentes módulos que se encargan de que el satélite opere correctamente desde el espacio.

Ground System. El Ground System es el segmento que se encarga del almacenamiento de los datos que se envían desde el OBSW, además, puede modificar su comportamiento utilizando telecomandos.

La Figura 3.3 ha sido realizada por Pablo Andreu Sedeño, jefe de proyecto del departamento de informática. Incluye los tres subsistemas que forman la misión, así como, los diferentes componentes de los mismos.

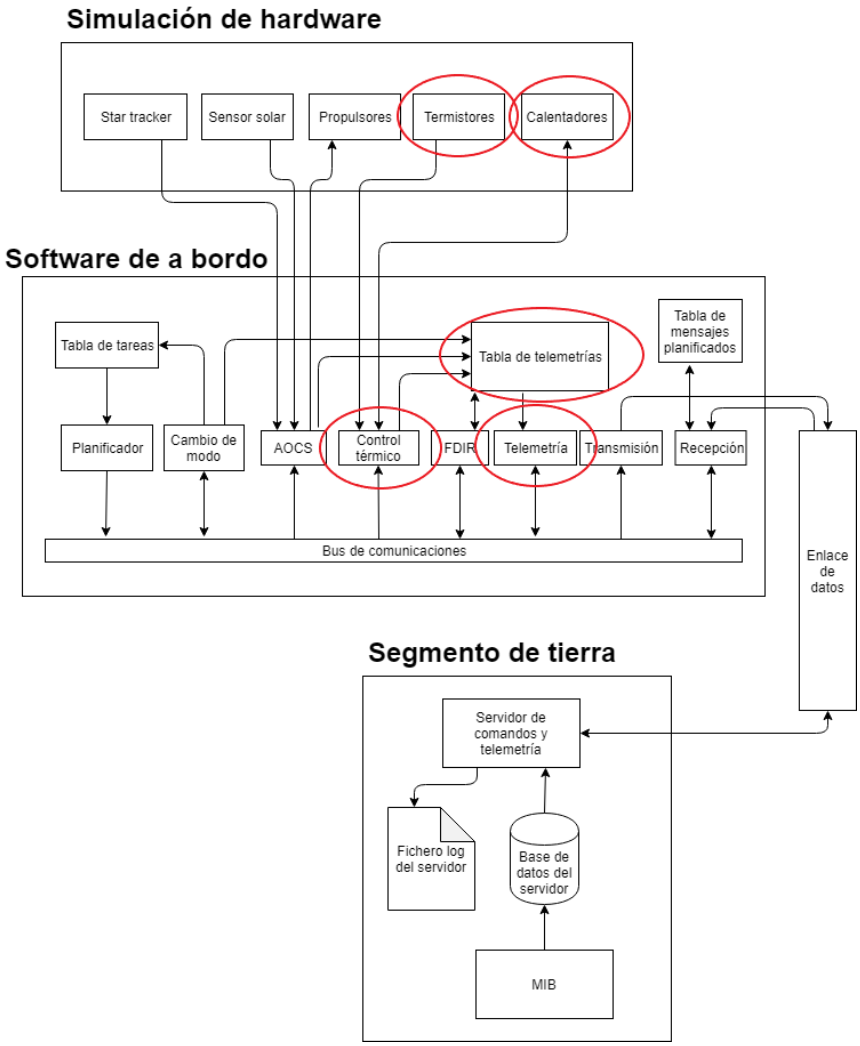


Figura 3.3: Arquitectura global del sistema

Arquitectura del sistema

Los componentes remarcados en la Figura 3.3 han sido implementados para este Trabajo de Fin de Grado y se explicarán en el Capítulo 4.

A continuación, se detallan los módulos que componen los tres subsistemas.

Componentes hardware

Star tracker. Es un dispositivo hardware, en concreto una cámara, que es capaz de calcular el cuaternión [1] del nanosatélite utilizando un eje de referencia. Las componentes que forman el cuaternión son calculadas comparando las posiciones de las estrellas en diferentes instantes de tiempo. Este componente será simulando utilizando Celestia [31].

[1] Notación matemática que se utiliza para representar la posición y la orientación de cuerpos en tres dimensiones

Sensor solar. Es un sensor que está formado por una matriz de celdas fotovoltaicas. Se obtendrá la posición solar según el ángulo de incidencia sobre cada celda.

Propulsores. Estos actuadores se encargan de modificar la posición del nanosatélite.

Termistores. Estos sensores se encarga de monitorizar la temperatura del sistema y alertar en caso de que se superen los umbrales definidos.

Calentadores. Estos actuadores se encargar de modificar la temperatura del sistema. Se encienden o apagan si es necesario calentar o enfriar el sistema para que todo funcione correctamente.

OBSW

Tabla de tareas. Esta tabla define el comportamiento del sistema. Indica las tareas que forman un modo, su periodo, su prioridad…

Tabla de telemetrías. Almacena todos los datos obtenidos por los sensores del sistema, así como, el control de error para cada uno de ellos. Además, almacena el estado (encendido/apagado) de los actuadores.

Tabla de comandos planificados. Almacena aquellos comandos que se ejecutarán en un instante de tiempo específico.

Planificador. Se encarga de leer la tabla de tareas según el modo en el que se opere y actuar en consecuencia. Es el encargado de activar o desactivar módulos según el modo de ejecución.

Cambio de modo. Se encarga de modificar la tabla de tareas que se debe ejecutar. Esta fase de la misión consta de dos modos de ejecución, son los siguientes:

Modo nominal. Modo de ejecución normal del sistema.

Modo seguro. Desactiva algunos módulos del OBSW y apunta al Sol para cargar baterías.

AOCS. Se encarga del control de posición orbital del nanosatélite. Lee los datos del Star tracker y modifica la posición del sistema activando o desactivando los propulsores.

Control térmico. Se encarga de la monitorización de la temperatura del sistema. Obtiene los datos de los termistores y modifica el comportamiento de los calentadores.

FDIR. Se encarga del control de errores del sistema. Verifica que los valores obtenidos por cada sensor se encuentran en los umbrales definidos. Distingue entre tres tipos de errores, son los siguientes:

Transitorios. Son errores puntuales, que pueden ser ocasionados por una mala lectura del sensor.

Permanentes. Es un conjunto de errores puntuales sobre un mismo sensor. Se alertará al módulo de telemetría para generar un paquete de error.

Fatales. Este tipo de errores que ponen en peligro la misión. Se generará un evento para cambiar a modo seguro, corrigiendo el comportamiento del sistema y evitando más fallos.

Telemetría. Se encarga de definir e implementar paquetes que serán enviados al Ground System. Estos paquetes pueden ser de dos tipos:

Periódicos. Se envía toda la información del sistema de manera periódica.

Puntuales. Se envía la información de un determinado componente del sistema cuando es requerido por el Ground System.

El envío y recepción de paquete entre módulos se realiza utilizando el Software Bus.

Transmisión. Se encarga del envío de paquetes de telemetría al Ground System utilizando el *data link*. La comunicación se realiza utilizando *sockets* UDP.

Recepción. Se encarga de recibir telecomandos desde el Ground System utilizando el *data link*. La comunicación se realiza utilizando *sockets* UDP.

Ground System

Servidor de comandos y telemetría. Se encarga de recibir los datos de telemetría del segmento espacial. Además, permite enviar telecomandos para modificar el comportamiento del OBSW. La comunicación se realiza utilizando *sockets* UDP.

Fichero log del servidor. Se encarga de almacenar en un fichero los datos recibidos del OBSW.

Base de datos del servidor. Se encarga de almacenar los datos de la misión definidos en el MIB.

MIB. Se encarga de la definición de valores para la misión. Estos valores son los siguientes:

- Telecomandos.
- Telemetría.
- Umbrales (máximos y mínimos) de cada sensor.

Todos los módulos y sensores se desarrollan sobre la última capa del sistema, ya que, son aplicaciones de usuario que hacen uso de los servicios que ofrece cFE.

El objetivo de este Trabajo de Fin de Grado es el diseño e implementación del módulo de telemetría y de la simulación software de los componentes térmicos.

3.2. Requisitos de usuario

Los requisitos de usuario indican la funcionalidad y las restricciones del sistema a desarrollar. Estos requisitos se han obtenido en reuniones con el jefe de proyecto, que a su vez, han sido definidos por el cliente final, SENER. Este tipo de requisitos se agrupan en dos bloques:

Requisitos de capacidad. Estos requisitos detallan la funcionalidad que debe realizar la solución final.

Requisitos de restricción. Estos requisitos imponen restricciones sobre los requisitos de capacidad.

Los requisitos de usuario se especifican utilizando una plantilla, véase la Figura 3.4 . Los atributos de la plantilla son los siguientes:

Identificador. Identifica unívocamente al requisito. Será XX-UR-YY, donde XX indica el tipo de requisito de usuario, ya sea de capacidad (CA) o de restricción (RE). YY indica el número de secuencia, comenzando por 01.

Descripción. Especificación del requisito en un lenguaje claro, conciso y no ambiguo.

Necesidad. Prioridad del requisito para el cliente (esencial, conveniente u opcional).

Prioridad. Prioridad del requisito para el desarrollador (alta, media o baja).

Estabilidad. Indica si el requisito se modifica durante el desarrollo (no cambia, cambiante o muy inestable).

Verificabilidad. Capacidad de comprobar la validez del requisito (alta, media o baja).

XX-UR-YY

Descripción:

Necesidad:

Prioridad:

Estabilidad:

Verificabilidad:

Figura 3.4: Plantilla de requisitos de usuario

Requisitos de capacidad

CA-UR-01

Descripción:	El módulo de telemetría debe enviar un paquete de forma periódica con todo el estado del sistema al módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-02

Descripción:	El módulo de telemetría debe enviar un paquete con toda la información térmica, control de errores y modo de ejecución cuando reciba una petición del módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-03

Descripción:	El módulo de telemetría debe enviar un paquete con toda la información posicional del nanosatélite, control de errores y modo de ejecución cuando reciba un comando del módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-04

Descripción:	El módulo de telemetría debe enviar un paquete con toda la información de posición respecto al Sol, control de errores y modo de ejecución cuando reciba una petición del módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-05

Descripción:	La tabla de telemetría debe almacenar los valores de monitorización del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-06

Descripción:	Los valores de la tabla de telemetría deben ser accesibles por todos los módulos del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-07

Descripción:	Los termistores deben realizar una lectura de la temperatura del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-08

Descripción:	La simulación de los calentadores debe provocar un aumento de la lectura de la temperatura interna del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-09

Descripción:	La monitorización térmica del sistema debe ser realizada por 3 termistores.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-10

Descripción:	Cada termistor debe tener asociado un calentador para aumentar la temperatura.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta

CA-UR-11

Descripción:	El control térmico se debe encargar de verificar que la temperatura se encuentra en los umbrales preestablecidos.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-12

Descripción:	El control térmico debe encender o apagar los calentadores según se necesite.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-13

Descripción:	Todos los paquetes deben incluir una marca temporal.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media

CA-UR-14

Descripción:	Se debe verificar la integridad de cada paquete.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media

CA-UR-15

Descripción:	Se pueden recibir mensajes del módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-16

Descripción:	Se pueden recibir mensajes del planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de restricción

RE-UR-01

Descripción:	El envío de paquetes con el resto de módulos se debe realizar utilizando el servicio bus software.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-02

Descripción:	Se debe utilizar el servicio de eventos para notificar acciones al sistema.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	Cambiante
Verificabilidad:	Alta

RE-UR-03

Descripción:	El sistema global debe incluir el módulo de telemetría.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-04

Descripción:	El sistema global debe incluir el módulo de control térmico.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-05

Descripción:	La implementación del sistema se debe realizar en C/C++.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-06

Descripción:	La implementación del sistema se debe realizar utilizando el *framework* cFE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-07

Descripción:	La solución completa debe ser funcional en un RTOS.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-08

Descripción:	La tarea de control térmico se debe ejecutar periódicamente cuando sea activada por el planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-09

Descripción:	La tarea de telemetría se debe ejecutar periódicamente cuando sea activada por el planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-10

Descripción:	El valor máximo de temperatura es de 223 K.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-11

Descripción:	El valor mínimo de temperatura es de 203 K.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-12

Descripción:	La tarea de telemetría no puede recibir mensajes que no sean del planificador y del módulo de comunicación.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-13

Descripción:	El módulo de telemetría tiene comunicación con el resto de módulos del satélite.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-14

Descripción:	El control térmico tiene comunicación con el resto de módulos del satélite.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-15

Descripción:	El módulo de telemetría se integrará con el resto de módulos de cFE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-16

Descripción:	El control térmico se integrará con el resto de módulos de cFE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

RE-UR-17

Descripción:	El control térmico no puede recibir mensajes que no sean del planificador y del módulo de comunicación.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

3.3. Casos de uso

Un diagrama de casos de uso es bastante útil durante el proceso de obtención de requisitos, puesto que, permite al cliente visualizar la funcionalidad que ofrece el sistema.

Cada caso de uso, describe el proceso que debe seguir un agente externo (actor) para ejecutar una determinada funcionalidad. Véase la Figura 3.6.

Los casos de uso se especifican utilizando la plantilla de la Figura 3.5. Los atributos que componen esta plantilla son los siguientes:

Identificador. Identifica unívocamente al caso de uso. Será UC-XX, donde XX indica el número de secuencia comenzando en 01.

Nombre. Descripción breve del caso de uso.

Actores. Agente externo que ejecuta el caso de uso.

Objetivo. Propósito del caso de uso.

Descripción. Pasos que debe seguir el actor para ejecutar el caso de uso.

Pre-condición. Condiciones previas que se deben cumplir para ejecutar el caso de uso.

Post-condición. Condiciones que se deben cumplir después de ejecutar el caso de uso.

UC-XX

Nombre:

Actores:

Objetivo:

Descripción:

Pre-condición:

Post-condición:

Figura 3.5: Plantilla de casos de uso

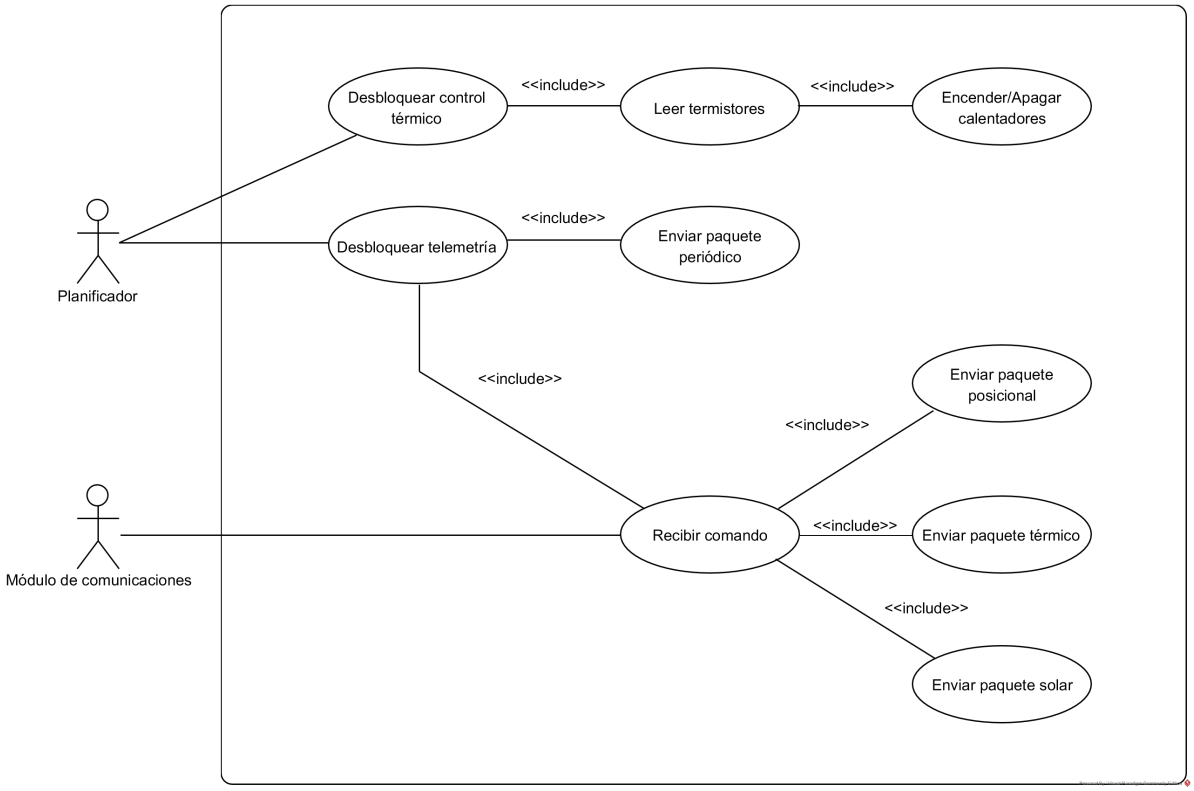


Figura 3.6: Diagrama UML de casos de uso

UC-01

Nombre:	Desbloquear telemetría
Actores:	Planificador
Objetivo:	Ejecutar la tarea de telemetría
Descripción:	El planificador envía un mensaje a la tarea de telemetría, que se desbloquea y comienza su ejecución.
Precondición:	▪ Estar suscrito al tipo de mensaje del planificador ▪ Comprobar si hay mensaje del planificador ▪ El atributo de identificación debe ser el nombre de la tarea de telemetría
Postcondición:	La tarea de telemetría se desbloquea y ejecuta

UC-02

Nombre:	Enviar paquete periódico
Actores:	Planificador
Objetivo:	Enviar paquete periódico al módulo de comunicación
Descripción:	Se envía un paquete con toda la información del sistema al módulo de comunicaciones.
Precondición:	▪ La tarea debe estar desbloqueada
Postcondición:	Se ha enviado el paquete periódico

UC-03

Nombre:	Recibir comando
Actores:	Planificador, Comunicaciones
Objetivo:	Recibir comando del módulo de comunicaciones
Descripción:	Se recibe un comando desde el módulo de comunicaciones en el que se indica que paquete puntual se debe enviar.
Precondición:	▪ La tarea debe estar desbloqueada ▪ El módulo de comunicaciones ha enviado un comando
Postcondición:	Se ha recibido el comando

UC-04

Nombre:	Enviar paquete posicional
Actores:	Planificador, Comunicaciones
Objetivo:	Enviar paquete posicional al módulo de comunicación
Descripción:	Se genera el paquete con toda la información de posición del satélite y se envía al módulo de comunicaciones.
Precondición:	▪ La tarea debe estar desbloqueada ▪ El módulo de comunicaciones ha enviado un comando ▪ El comando solicita el paquete posicional
Postcondición:	Se ha enviado el paquete posicional

UC-05	
Nombre:	Enviar paquete térmico
Actores:	Planificador, Comunicaciones
Objetivo:	Enviar paquete térmico al módulo de comunicación
Descripción:	Se genera el paquete con toda la información térmica del satélite y se envía al módulo de comunicaciones.
Pre-condición:	▪ La tarea debe estar desbloqueada ▪ El módulo de comunicaciones ha enviado un comando ▪ El comando solicita el paquete térmico
Post-condición:	Se ha enviado el paquete térmico

UC-06	
Nombre:	Enviar paquete solar
Actores:	Planificador, Comunicaciones
Objetivo:	Enviar paquete solar al módulo de comunicación
Descripción:	Se genera el paquete con toda la información de posición del satélite respecto al Sol y se envía al módulo de comunicaciones.
Pre-condición:	▪ La tarea debe estar desbloqueada ▪ El módulo de comunicaciones ha enviado un comando ▪ El comando solicita el paquete solar
Post-condición:	Se ha enviado el paquete solar

UC-07

Nombre:	Desbloquear control térmico
Actores:	Planificador
Objetivo:	Ejecutar el control térmico
Descripción:	El planificador envía un mensaje al control térmico, que se desbloquea y comienza su ejecución.
Precondición:	▪ Estar suscrito al tipo de mensaje del planificador ▪ Comprobar si hay mensaje del planificador ▪ El atributo de identificación debe ser el nombre del control térmico
Postcondición:	El control térmico se desbloquea y ejecuta

UC-08

Nombre:	Leer termistores
Actores:	Planificador
Objetivo:	Simular una lectura de la temperatura del sensor
Descripción:	Se simula una lectura de temperatura del sistema.
Precondición:	▪ La tarea de control térmico debe estar desbloqueada
Postcondición:	Se ha leído la temperatura del sensor

UC-09

Nombre:	Encender/Apagar calentadores
Actores:	Planificador
Objetivo:	Cambiar el estado de los calentadores
Descripción:	Se cambia el estado de los calentadores si la temperatura del sistema excede los límites preestablecidos.
Precondición:	▪ La tarea de control térmico debe estar desbloqueada ▪ El valor de temperatura leído no está en el rango definido
Postcondición:	Se cambia el estado del calentador

3.4. Requisitos de software

En esta sección se incluye la especificación de requisitos software. Estos requisitos se han obtenido de los requisitos de usuario definidos en la Sección 3.2 y describen lo que el analista ha entendido de dicho proceso.

Los requisitos software se dividen en dos bloques:

Requisitos funcionales. Especifican las características funcionales del software.

Requisitos no funcionales. Especificaciones complementarias que no aportan funcionalidad.

Los requisitos de software se especifican utilizando la plantilla de la Figura 3.7. Los atributos de la plantilla son los siguientes:

Identificador. Identifica unívocamente al requisito, será XX-SR-YY, donde:

- XX puede ser F (requisito funcional) o NF (requisito no funcional).
- YY es el número de secuencia iniciado en 01.

Descripción. Especificación del requisito en un lenguaje claro, conciso y no ambiguo.

Necesidad. Prioridad del requisito para el cliente (esencial, conveniente u opcional).

Prioridad. Prioridad del requisito para el desarrollador (alta, media o baja).

Estabilidad. Indica si el requisito se modifica durante el desarrollo (no cambia, cambiante o muy inestable).

Verificabilidad. Capacidad de comprobar la validez del requisito (alta, media o baja).

Origen. Referencia a los requisitos de usuario que dieron lugar al requisito.

XX-SR-YY
Descripción:
Necesidad:
Prioridad:
Estabilidad:
Verificabilidad:
Origen:

Figura 3.7: Plantilla de requisitos software

Requisitos funcionales

F-SR-01

Descripción: Se enviará una estructura al software bus con los siguientes campos:

- `uint8 TlmHeader[TLM_HDR_SIZE]`
- `int32_t thermistors[3]`
- `int32_t position[4]`
- `int32_t solarquaternion[4]`
- `int32_t voltage`
- `int32_t flags`

Necesidad: Esencial

Prioridad: Alta

Estabilidad: Cambiante

Verificabilidad: Alta

Origen: CA-UR-01, CA-UR-13, CA-UR-14

F-SR-02

Descripción: Se enviará una estructura al software bus con los siguientes campos:

- `uint8 TlmHeader[TLM_HDR_SIZE]`
- `int32_t thermistors[3]`
- `int32_t flags`

Necesidad: Esencial

Prioridad: Alta

Estabilidad: Cambiante

Verificabilidad: Alta

Origen: CA-UR-02, CA-UR-13, CA-UR-14

F-SR-03

Descripción: Se enviará una estructura al software bus con los siguientes campos:

- `uint8 TlmHeader[TLM_HDR_SIZE]`
- `int32_t position[4]`
- `int32_t flags`

Necesidad: Esencial

Prioridad: Alta

Estabilidad: Cambiante

Verificabilidad: Alta

Origen: CA-UR-03, CA-UR-13, CA-UR-14

F-SR-04

Descripción:	Se enviará una estructura al software bus con los siguientes campos: ▪ `uint8 TlmHeader[TLM_HDR_SIZE]` ▪ `int32_t voltage` ▪ `int32_t solarquaternion[4]` ▪ `int32_t flags`
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-04, CA-UR-13, CA-UR-14

F-SR-05

Descripción:	Como parte de la definición del planificador (componente externo), la estructura del paquete del planificador es la siguiente: ▪ `uint8 SchHeader[SCH_HDR_SIZE]` ▪ `char task[20]`
Necesidad:	Esencial
Prioridad:	Baja
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-16

F-SR-06

Descripción:	Como parte de la definición del módulo de comunicaciones (componente externo), la estructura del paquete comandos es la siguiente: ▪ `uint8 CmdHeader[CMD_HDR_SIZE]` ▪ `int32_t command_identifier` ▪ `int32_t associated_value` ▪ `int32_t execution_time`
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-15

F-SR-07

Descripción:	La tabla de telemetría almacenará los datos de monitorización del sistema (lectura de sensores, modo de ejecución, estado de actuadores, etc).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Media
Origen:	CA-UR-05,CA-UR-06

F-SR-08

Descripción:	El control térmico almacena los valores de los termistores en la tabla de telemetría.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-06

F-SR-09

Descripción:	El control térmico activa el calentador correspondiente si el valor leído del termistor asociado desciende del umbral mínimo.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-08,CA-UR-11, CA-UR-12

F-SR-10

Descripción:	El control térmico desactiva el calentador correspondiente si el valor leído del termistor asociado supera el umbral máximo.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-08, CA-UR-11, CA-UR-12

F-SR-11

Descripción:	Se incluirá un *checksum* para propósitos de comprobación de integridad en los siguientes casos: ▪ Paquete periódico ▪ Paquete puntual (Posicional, térmico y solar)
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-14

F-SR-12

Descripción:	Se incluirá un *timestamp* para conocer la fecha de creación de paquete en los siguientes casos: ▪ Paquete periódico ▪ Paquete puntual (Posicional, térmico y solar)
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	CA-UR-13

F-SR-13

Descripción:	Se simulan 3 termistores encargados de leer la temperatura del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-07, CA-UR-09

F-SR-14

Descripción:	Cada termistor tiene un calentador asociado que simula un aumento en la temperatura del sistema.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Cambiante
Verificabilidad:	Alta
Origen:	CA-UR-10

F-SR-15

Descripción:	Los datos monitorizados por el sistema deben estar una región de memoria que sea accesible por el resto de módulos (memoria compartida).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-05

Requisitos no funcionales

NF-SR-01

Descripción:	El envío y recepción de mensajes se realiza utilizando Software Bus.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-01

NF-SR-02

Descripción:	Se notificará al sistema el inicio de cada módulo.
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	Cambiante
Verificabilidad:	Media
Origen:	RE-UR-02

NF-SR-03

Descripción:	Se notificará al sistema el envío de cada paquete.
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	Cambiante
Verificabilidad:	Media
Origen:	RE-UR-02

NF-SR-04

Descripción:	El módulo de telemetría está integrado con el resto del sistema, ver Figura 3.3.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-03,RE-UR-15

NF-SR-05

Descripción:	El control térmico está integrado con el resto del sistema, ver Figura 3.3.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-04,RE-UR-16

NF-SR-06

Descripción:	El sistema se desarrollan en C utilizando el *framework* cFE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Baja
Origen:	RE-UR-05,RE-UR-06

NF-SR-07

Descripción:	Cada módulo del sistema es una aplicación de cFE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-06

NF-SR-08

Descripción:	El sistema es funcional en RTEMS.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-07

NF-SR-09

Descripción:	La tarea de telemetría se ejecuta sólo cuando recibe el mensaje del planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-09

NF-SR-10

Descripción:	El control térmico se ejecuta sólo cuando recibe el mensaje del planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-08

NF-SR-11

Descripción:	El sistema no supera los límites establecidos de temperatura.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-10,RE-UR-11

NF-SR-12

Descripción:	El módulo de telemetría solo recibe mensajes del planificador y del módulo de comunicaciones.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-12

NF-SR-13

Descripción:	El control térmico solo recibe mensajes del planificador.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-17

NF-SR-14

Descripción:	El módulo de telemetría se comunica con todo el sistema utilizando Software Bus.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-13

NF-SR-15

Descripción:	El control térmico se comunica con todo el sistema utilizando Software Bus.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-14

3.5. Diseño de la solución

En esta sección se detalla el sistema realizado. En primer lugar, se realiza el diseño arquitectónico de los módulos a desarrollar. Seguidamente, se realizará el diseño dinámico de los mismos en el que se puede comprobar su funcionalidad mediante diagramas de interacción.

Diseño estático

A continuación se especifican los componentes que forman el sistema, para ello, se utiliza la plantilla de la Figura 3.8. Los atributos de esta plantilla son los siguientes:

Identificador. Nombre del componente.

Rol. Papel del componente en el sistema.

Dependencias. Componentes que dependen de este.

Descripción. Explicación del componente.

Datos. Valores de entrada [in] y salida [out] del componente.

Recursos. Recursos que utiliza el componente.

Origen. Requisitos software que dieron lugar al componente.

Identificador	
Rol:	
Dependencias:	
Descripción:	
Datos:	n/a
Recursos:	n/a
Origen:	

Figura 3.8: Plantilla de componentes

En la Figura 3.9 se puede observar el diagrama de los componentes. Los componentes externos que interaccionan con los propios son de color rojo.

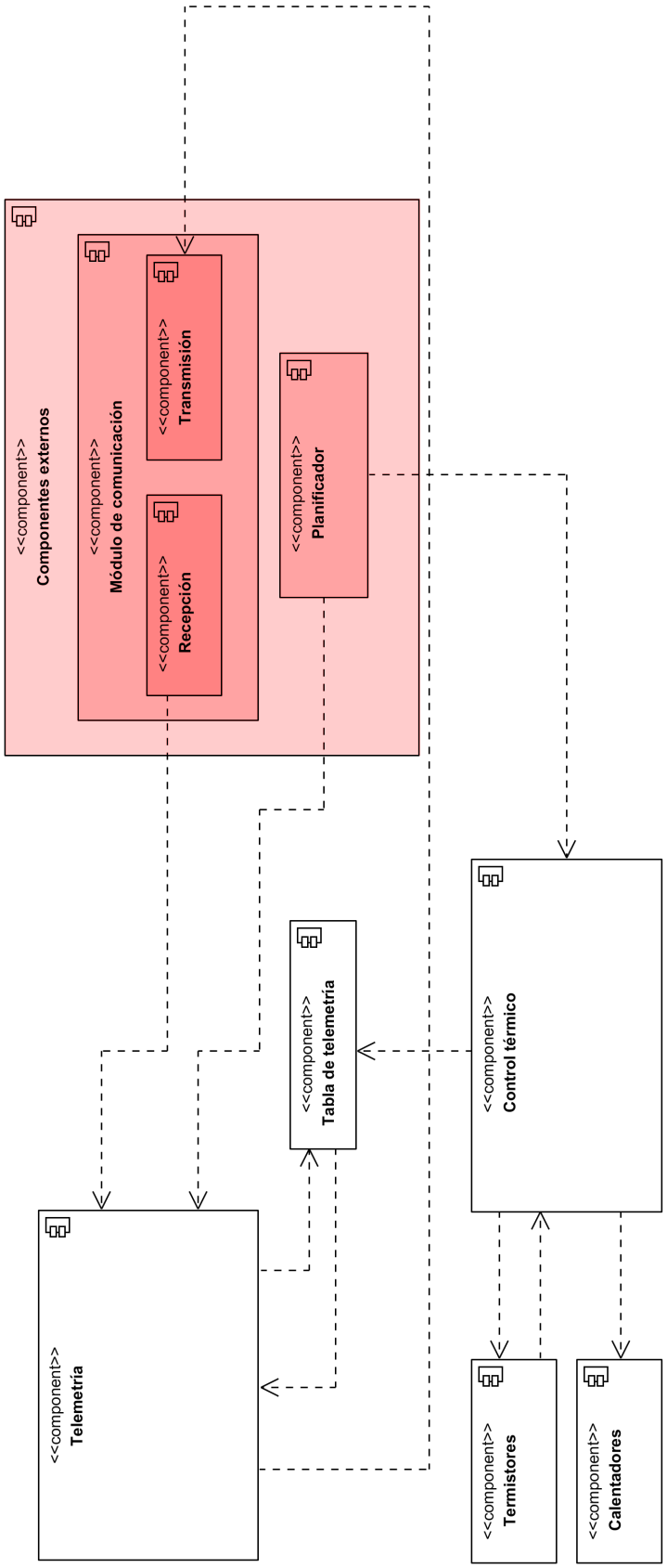


Figura 3.9: Diagrama de componentes

Telemetría

Rol:	Monitorización del sistema y envío de paquetes
Dependencias:	Planificador, Comunicaciones
Descripción:	El módulo de telemetría se encarga de monitorizar el estado del sistema, generar paquetes y enviar al Software Bus.
Datos:	▪ [in]: Mensaje del planificador y comando de comunicaciones ▪ [out]: Paquete periódico. Paquete posicional. Paquete térmico. Paquete solar
Recursos:	n/a
Origen:	F-SR-01,F-SR-02,F-SR-03,F-SR-04,F-SR-05,F-SR-06,F-SR-11,F-SR-12

Control térmico

Rol:	Se encarga del control térmico del sistema.
Dependencias:	Planificador
Descripción:	El control térmico se encarga de verificar que la temperatura del sistema se encuentra en los umbrales definidos. Se activan o desactivan los calentadores para modificar la temperatura.
Datos:	▪ [in]: Mensaje del planificador ▪ [out]: Modificar valores en tabla de telemetría. Encender o apagar calentadores
Recursos:	n/a
Origen:	F-SR-08,F-SR-09,F-SR-10

Tabla de telemetría

Rol:	Almacena los datos de monitorización
Dependencias:	Módulos de cFE
Descripción:	La tabla de telemetría almacena todos los valores de monitorización del sistema para que sean accesible por diversos módulos.
Datos:	▪ [in]: Valores de monitorización ▪ [out]: -
Recursos:	n/a
Origen:	F-SR-07,F-SR-15

Termistores

Rol:	Simula una lectura de temperatura del sistema
Dependencias:	Control térmico
Descripción:	Los termistores simulan una lectura de temperatura del sistema. Envían los valores al control térmico.
Datos:	▪ [in]: Solicitud de lectura ▪ [out]: Lectura de temperatura
Recursos:	n/a
Origen:	F-SR-13

Calentadores

Rol:	Simula un aumento en la temperatura del sistema
Dependencias:	Control térmico
Descripción:	Los calentadores simulan un aumento de temperatura del sistema. Envían su estado al control térmico.
Datos:	▪ [in]: Solicitud de cambio de estado de calentadores ▪ [out]: Estado del calentador
Recursos:	n/a
Origen:	F-SR-14

Diseño dinámico

Para poder observar la interacción de los componentes definidos previamente, se han definido los siguientes diagramas de interacción:

Envío paquete periódico. En este diagrama se detalla la interacción existente entre el módulo de telemetría, la tabla de telemetría, el planificador y el módulo de transmisión para el envío del paquete periódico con toda la monitorización del sistema. Véase la Figura 3.10.

Envío de paquete puntual En este diagrama se detalla la interacción existente entre el módulo de telemetría, la tabla de telemetría, el planificador, el módulo de transmisión y el módulo de recepción para el envío del paquete puntual con la información solicitada por el segmento terrestre. Véase la Figura 3.11.

Control normal. En este diagrama se detalla la interacción entre el control térmico, los termistores, la tabla de telemetría y el planificador para realizar una lectura de la temperatura del sistema. Además, se almacenan los valores leídos en la tabla de telemetría para que sean accesibles por cualquier módulo. Véase la Figura 3.12.

Cambio de estado de calentadores. En este diagrama se detalla la interacción entre el control térmico, los termistores, los calentadores, la tabla de telemetría y el planificador para realizar una lectura de la temperatura del sistema. Se cambia el estado de los calentadores, ya que, se superan los umbrales establecidos. Además, se almacenan los valores leídos en la tabla de telemetría para que sean accesibles por cualquier módulo. Véase la Figura 3.13.

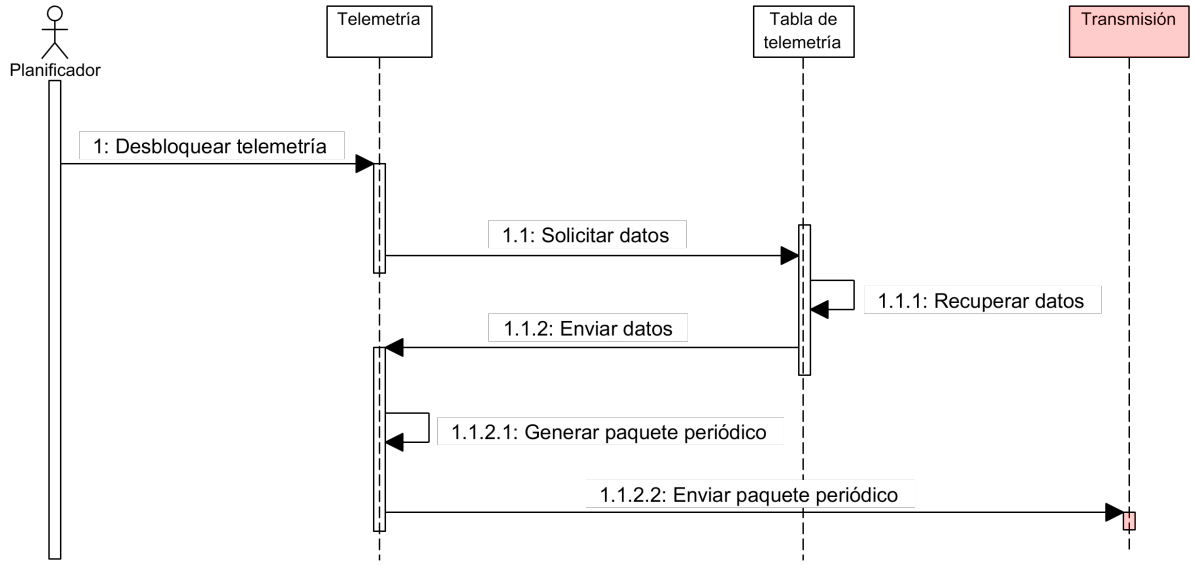


Figura 3.10: Diagrama de secuencia de telemetría I (enviar paquete periódico)

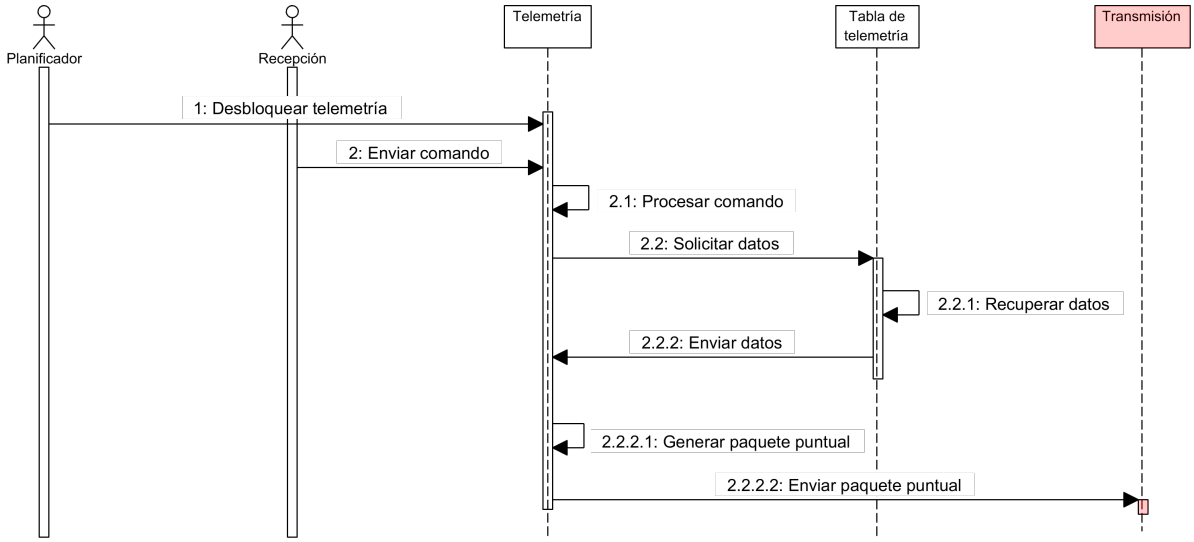


Figura 3.11: Diagrama de secuencia de telemetría II (enviar paquete puntual)

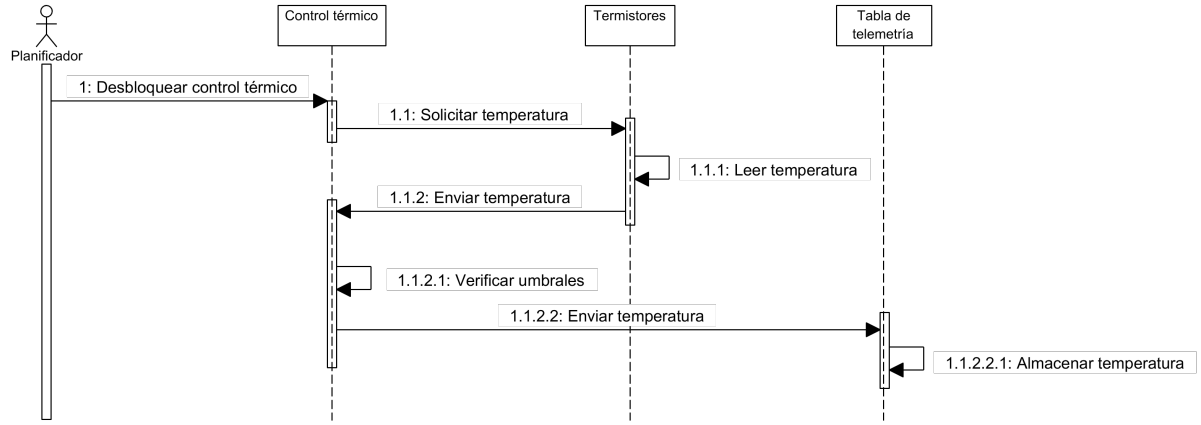


Figura 3.12: Diagrama de secuencia control térmico I (dentro de rango)

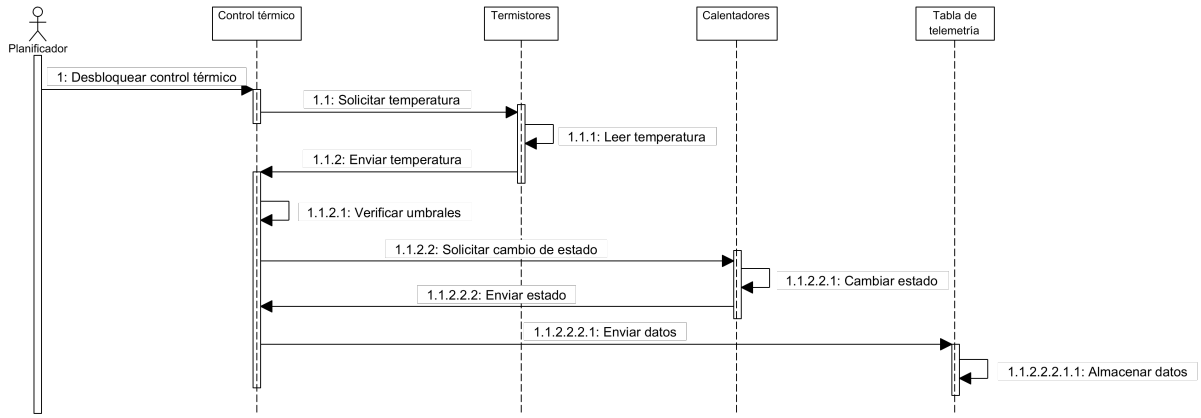


Figura 3.13: Diagrama de secuencia control térmico II (fuera de rango)

3.6. Matrices de trazabilidad

Esta sección incluye las matrices de trazabilidad entre requisitos de usuario y requisitos software, además de la matriz entre requisitos funcionales y componentes.

La matriz de trazabilidad entre requisitos de capacidad - requisitos funcionales se encuentra en la Figura 3.14, mientras que, la matriz de trazabilidad requisitos de restricción - requisitos no funcionales se encuentra en la Figura 3.15. Finalmente, la matriz de trazabilidad requisitos funcionales - componentes se encuentra en la Figura 3.16.

Matriz de trazabilidad CA-UR / F-SR

	CA-UR-01	CA-UR-02	CA-UR-03	CA-UR-04	CA-UR-05	CA-UR-06	CA-UR-07	CA-UR-08	CA-UR-09	CA-UR-10	CA-UR-11	CA-UR-12	CA-UR-13	CA-UR-14	CA-UR-15	CA-UR-16
F-SR-01	•											•	•			
F-SR-02		•										•	•			
F-SR-03			•									•	•			
F-SR-04				•								•	•			
F-SR-05																•
F-SR-06														•		
F-SR-07					•	•										
F-SR-08						•										
F-SR-09								•			•	•				
F-SR-10								•			•	•				
F-SR-11													•			
F-SR-12												•				
F-SR-13							•		•							
F-SR-14										•						
F-SR-15					•											

Figura 3.14: Matriz de trazabilidad requisitos de capacidad-requisitos funcionales

Matriz de trazabilidad RE-UR / NF-SR

	RE-UR-01	RE-UR-02	RE-UR-03	RE-UR-04	RE-UR-05	RE-UR-06	RE-UR-07	RE-UR-08	RE-UR-09	RE-UR-10	RE-UR-11	RE-UR-12	RE-UR-13	RE-UR-14	RE-UR-15	RE-UR-16	RE-UR-17
NF-SR-01	•																
NF-SR-02		•															
NF-SR-03		•															
NF-SR-04			•												•		
NF-SR-05				•												•	
NF-SR-06					•	•											
NF-SR-07						•											
NF-SR-08							•										
NF-SR-09									•								
NF-SR-10								•									
NF-SR-11										•	•						
NF-SR-12												•					
NF-SR-13																	•
NF-SR-14													•				
NF-SR-15														•			

Figura 3.15: Matriz de trazabilidad requisitos de restricción-requisitos no funcionales

Matriz de trazabilidad F-SR/Componentes

	F-SR-01	F-SR-02	F-SR-03	F-SR-04	F-SR-05	F-SR-06	F-SR-07	F-SR-08	F-SR-09	F-SR-10	F-SR-11	F-SR-12	F-SR-13	F-SR-14	F-SR-15
Calentadores														•	
Control térmico								•	•	•					
Tabla de telemetría							•								•
Telemetría	•	•	•	•	•	•					•	•			
Termistores													•		

Figura 3.16: Matriz de trazabilidad componentes - requisitos funcionales

Capítulo 4

Implementación

En este capítulo se incluyen los detalles de implementación más relevantes del módulo de telemetría y del simulador térmico. El proceso de instalación del sistema, diseño de módulos y ejecución queda recogido en el manual de usuario, véase el Apéndice C.

4.1. Módulo de telemetría

El módulo de telemetría es el encargado de monitorizar el comportamiento del sistema y generar los paquetes correspondientes que recibirá el Ground System. Todos los paquetes de telemetría tienen una estructura similar, véase la Figura 4.1, únicamente se diferencian en los datos de usuario.

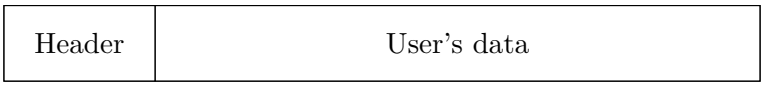


Figura 4.1: Estructura de paquetes

Para evitar errores con números en coma flotante y el overhead en arquitecturas sin soporte hardware para IEEE 754, toda la información se codificará en `int32_t` con una precisión de 10^3, es decir, el valor en coma flotante se multiplicará por 1000 y se realizará una coerción al tipo expuesto anteriormente. Esta transformación se realiza para evitar pérdidas de precisión y porque muchos sistemas empotrados no soportan aritmética flotante en hardware.

Toda la comunicación entre módulos se realiza utilizando el servicio Software Bus de cFE.

Software bus

Software Bus es un servicio que ofrece cFE. Este servicio permite la comunicación entre los módulos del sistema utilizando mensajes. Se utiliza un modelo *Publish–Subscribe* en el que todos los mensajes son accesibles desde el Software Bus y cada módulo se suscribe a un determinado tipo de mensajes utilizando una *pipe* FIFO [1] de cFE. Véase la Figura 4.2.

[1] First in, first out

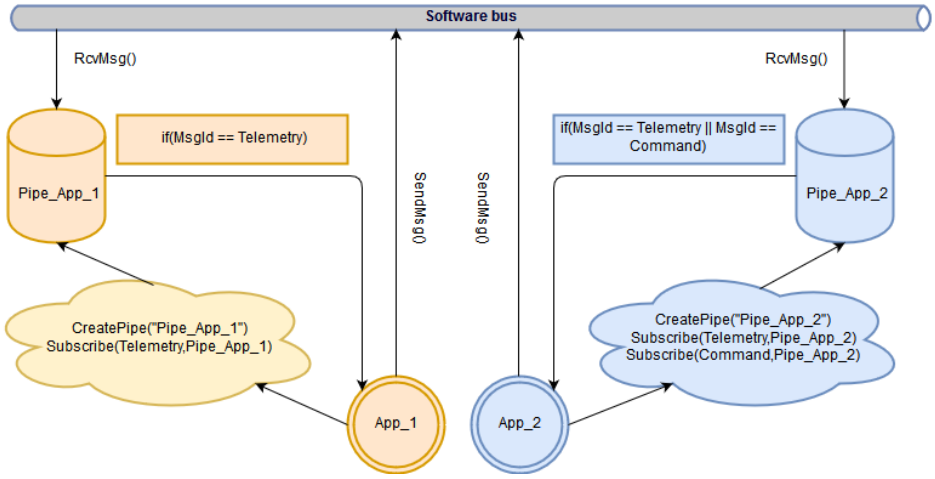


Figura 4.2: Funcionamiento de Software Bus

Este módulo es esencial para el desarrollo del módulo de telemetría, ya que, permite enviar los paquetes con toda la información del sistema al módulo de comunicación que los enviará al Ground System utilizando *sockets* UDP.

Los datos que se envían en cada paquete son recuperados desde la tabla de telemetría, que es accesible por todos los módulos del sistema. Todos los paquetes constan de dos partes fundamentales, son las siguientes:

Cabecera. Es la cabecera del paquete, en la que se incluye la información más relevante del mismo (identificador, *timestamp*, *checksum*, etc).

Datos de usuario. Contiene todos aquellos datos que el usuario decide que se enviarán al Ground System (monitorización de sensores, estado de actuadores, modo de ejecución, etc).

Los atributos que se incluyen en los paquetes de telemetría son los siguientes:

`uint8 TlmHeader[TLM_HDR_SIZE].` Cabecera del paquete que incluye el identificador de mensaje (MID), *timestamp*, etc.

`int32_t thermistors[3].` Incluye el valor leído por los tres termistores.

`int32_t position[4].` Información de posición y orientación (cuaternión) del nanosatélite.

`int32_t solarposition[4].` Cuaternión del nanosatélite respecto al Sol.

`int32_t voltage.` Voltaje producido por las células fotoeléctricas de la placa solar.

`int32_t flags.` Incluye el control de errores e información del sistema a nivel de bit, véase la Subsección 4.1.

Los diferentes tipos de paquetes que envía el módulo de telemetría son los siguientes:

Paquete periódico Este paquete contiene toda la información disponible del sistema (valores de sensores, estado de actuadores, modo de ejecución, etc) y se envía periódicamente al Software Bus cuando el planificador activa la tarea de telemetría. Véase el Listado 4.1.

Paquete térmico. Este paquete contiene toda la información relativa al control térmico (valores de los termistores y *flags* de control). Se envía cuando el módulo de telemetría recibe un comando del módulo de comunicación solicitando esta información. Véase el Listado 4.2.

Paquete posicional. Este paquete contiene toda la información relativa a la posición del nanosatélite (valores de posición obtenidos del startracker y *flags* de control). Se envía cuando el módulo de telemetría recibe un comando del módulo de comunicación solicitando esta información. Véase el Listado 4.3.

Paquete solar. Este paquete contiene toda la información relativa al sensor solar (valores de posición respecto al Sol, voltaje medio del sistema y *flags* de control). Se envía cuando el módulo de telemetría recibe un comando del módulo de comunicación solicitando esta información. Véase el Listado 4.4.

```
typedef struct {
  uint8 TlmHeader[TLM_HDR_SIZE];
  int32_t thermistors[3];
  int32_t position[4];
  int32_t solarposition[4];
  int32_t voltage;
  int32_t flags;
} OS_PACK telemetry_packet;
#define TELEMETRY_PACKET_LNGTH   sizeof ( telemetry_packet )
#define TELEMETRY_PACKET_MID          0x1890
```

Listado 4.1: Paquete periódico

```
typedef struct {
  uint8 TlmHeader[TLM_HDR_SIZE];
  int32_t thermistors[3];
  int32_t flags;
} OS_PACK thermic_packet;
#define THERMIC_PACKET_LNGTH   sizeof ( thermic_packet )
#define THERMIC_PACKET_MID          0x1891
```

Listado 4.2: Paquete térmico

```
typedef struct {
  uint8 TlmHeader[TLM_HDR_SIZE];
  int32_t quaternion[4];
  int32_t flags;
} OS_PACK positional_packet;
#define SOLAR_PACKET_LNGTH   sizeof ( positional_packet )
#define SOLAR_PACKET_MID          0x1892
```

Listado 4.3: Paquete posicional

```
typedef struct {
  uint8 TlmHeader[TLM_HDR_SIZE];
  int32_t solarquaternion[4];
  int32_t voltage;
  int32_t flags;
} OS_PACK solar_packet;
#define SOLAR_PACKET_LNGTH   sizeof ( solar_packet )
#define SOLAR_PACKET_MID          0x1893
```

Listado 4.4: Paquete solar

Asignación de flags

31	30	29	28	···	1	0

Figura 4.3: Esquema del miembro *flags*

Para optimizar el tamaño de los paquetes, se empaquetarán varios indicadores que permiten conocer el estado de diversos componentes del sistema. Se utilizará una variable del tipo `int32_t` en el que cada bit se corresponde con un determinado componente o aspecto del sistema, véase la Tabla 4.1.

Los primeros 24 bits se utilizan para la información de los sensores y actuadores. Actualmente, solo se utilizan 13 bits, los bits restantes se reservan para nuevos sensores y actuadores que complementen la misión.

Los 8 bits restantes de la variable se utilizan para control del sistema. De manera análoga, se reservan 7 bits que se pueden utilizar para controlar otros parámetros del sistema. El bit 24 indica el modo de ejecución del sistema, que puede ser:

Nominal. Modo de ejecución por defecto del sistema. Se realizan todas las tareas que incluye el sistema. Todos los sensores y actuadores están operativos.

Seguro. Este modo de ejecución se activa cuando existe algún fallo importante en el sistema. Se desactivan todas las tareas, sensores y actuadores que no son esenciales para garantizar la integridad del sistema. El nanosatélite realiza un apuntamiento al Sol para cargar su batería y se espera a recibir un comando del Ground System para reiniciar el OBSW.

Bit	Característica	Descripción	Bit a 0	Bit a 1
0	Termistor 1	Indica si hay error en el primer termistor	Sin error	Error
1	Termistor 2	Indica si hay error en el segundo termistor	Sin error	Error
2	Termistor 3	Indica si hay error en el tercer termistor	Sin error	Error
3	Termistor global	Indica si hay dos errores o más en los termistores	Sin error	Error
4	Calentador 1	Indica el estado del primer calentador	Apagado	Encendido
5	Calentador 2	Indica el estado del segundo calentador	Apagado	Encendido
6	Calentador 3	Indica el estado del tercer calentador	Apagado	Encendido
7	Sensor solar 1	Indica si hay error en la primera fila del panel solar	Sin error	Error
8	Sensor solar 2	Indica si hay error en la segunda fila del panel solar	Sin error	Error
9	Sensor solar 3	Indica si hay error en la tercera fila del panel solar	Sin error	Error
10	Sensor solar global	Indica si hay dos errores o más en el panel solar	Sin error	Error
11	Propulsor 1	Indica el estado del primer propulsor	Apagado	Encendido
12	Propulsor 2	Indica el estado del segundo propulsor	Apagado	Encendido
13 - 23	Reservado	Disponible para futuros sensores/actuadores	-	-
24	Modo de ejecución	Indica el modo de ejecución	Nominal	Seguro
25-31	Reservado	Disponible para funciones futuras	-	-

Tabla 4.1: Bits del miembro *flags*

Funcionamiento del sistema

```
Data: Valores de monitorización del sistema
Result: Envío de paquete al Software Bus
while telemetríaDesbloqueada do
    generarPaquetePeriodico();
    enviarPaquetePeriodico();
    if comandosPendientes then
        switch comandoPendiente do
            case comandoTermico do
                generarPaqueteTermico();
                enviarPaqueteTermico();
            end
            case comandoPosicion do
                generarPaquetePosicional();
                enviarPaquetePosicional();
            end
            case comandoSolar do
                generarPaqueteSolar();
                enviarPaqueteSolar();
            end
        end
    end
end
```

Algorithm 1: Algoritmo telemetría

Al comenzar la ejecución del módulo, se crean aquellas estructuras que son necesarias durante el ciclo de vida (mensajes, *pipes*, etc). Se registra el módulo en el sistema cFE utilizando el servicio Executive y se crea un evento utilizando el servicio Event para informar al sistema que el módulo va a comenzar su ejecución.

Cuando se haya realizado este proceso, la tarea entra en un bucle infinito que se ejecutará hasta que se apague o reinicie el sistema. En ese momento, se realizará la llamada bloqueante *receive()* que espera a que llegue un mensaje del planificador antes de continuar con la ejecución de su *payload*.

Si el planificador desbloquea a la tarea, se accede a la tabla de telemetría para obtener todos los valores de monitorización, se crea el paquete periódico y se asigna toda la información. Antes de enviar este paquete, se genera un *checksum* que permita verificar su integridad, además se establece un *timestamp* para controlar su fecha de creación. Finalmente, se envía el paquete utilizando Software Bus y se envía un evento utilizando el servicio Event.

Una vez se ha mandado el paquete periódico, se vuelve ha realizar la llamada *receive()*, pero esta vez no es bloqueante, para recibir mensajes del módulo de comunicación indicando el paquete que se debe enviar. Se evalúa el atributo correspondiente que contiene un identificador unívoco para cada paquete, véase la Tabla 4.2.

De manera análoga al paquete periódico, se obtienen aquellos datos que se van a enviar, se crea el paquete y se asigna la información. Seguidamente, se crea su *checksum* y se asigna

ID	Tipo de paquete
0	Paquete térmico
1	Paquete posicional
2	Paquete solar

Tabla 4.2: Tipos de paquetes

su *timestamp*. Finalmente, se envía el paquete al Software Bus y se notifica con su evento correspondiente.

4.2. Simulador térmico

El simulador térmico consiste en la simulación software de aquellos componentes hardware relacionados con el control y gestión de la temperatura del nanosatélite. Consta de tres componentes, son los siguientes:

Termistores. Estos receptores se encargan de obtener la temperatura a la que se encuentra el sistema.

Calentadores. Estos actuadores se encargan de modificar la temperatura del sistema para evitar errores críticos.

Control térmico. Es el módulo encargado de controlar la temperatura del sistema. Se encarga de obtener los datos térmicos a través de los termistores, comprueba que se encuentran en los umbrales definidos y activa o desactiva los calentadores para modificar la temperatura.

Termistores

De acuerdo con los requisitos obtenidos en el Capítulo 3, la temperatura a la que debe funcionar el sistema se debe encontrar entre $203K$ y $223K$ asumiendo un periodo de 300 segundos. Para realizar esta simulación, se ha implementado una onda sinusoidal, véase la Ecuación 4.1.

$$T(t) = \mathcal{C} + A * sin(2\Pi \upsilon t + \varphi) \tag{4.1}$$

A. Amplitud de la onda.

υ. Frecuencia de la onda.

φ. Constante de fase.

Se obtendrá una función sinusoidal con un periodo de 300 segundos que no supera los umbrales establecidos, véase la Figura 4.4.

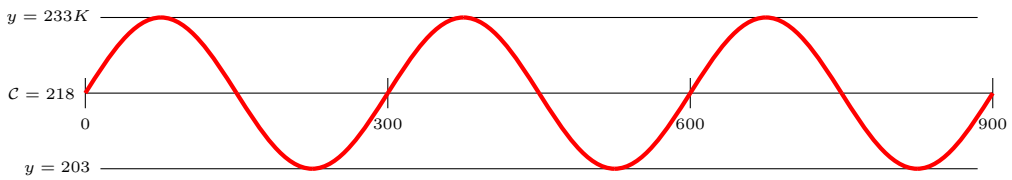


Figura 4.4: Función sinusoidal de temperatura

Calentadores

Los calentadores se encargan de simular un aumento en la temperatura del sistema. Son activados por el control térmico cuando la temperatura es inferior a $203K$ y se desactivan cuando alcanzan el umbral de $233K$. Cada vez que se encienden o apagan, el control térmico se encarga de almacenar su estado en la tabla de telemetría. Si está apagado se representa con el valor 0, si está encendido toma el valor 1.

Control térmico

```
Data: Valores de temperatura del sistema
Result: Valores de temperatura del sistema y estado de
        los calentadores
while controlTermicoDesbloqueado do
    lecturaTermistores();
    for termistores do
        if temperatura >MAX then
            apagarCalentador();
        end
        if temperatura <MIN then
            encenderCalentador();
        end
    end
    actualizarTablaTelemetria();
end
```

Algorithm 2: Algoritmo control térmico

El control térmico es el encargado de verificar que la temperatura del sistema se encuentra en el rango de valores establecido. Para ello, obtiene los valores de la tabla de telemetría, los evalúa y modifica el estado de los calentadores si es necesario.

Para lograr una simulación más real que permita comprobar el correcto funcionamiento de los calentadores y del módulo FDIR, se añade un valor de ruido a la lectura de cada termistor, véase la Ecuación 4.2. De esta forma, se consigue sobrepasar los umbrales preestablecidos. Por tanto, se modifica el estado de los calentadores y el módulo de detección de fallos puede comprobar su funcionalidad.

$$T_i = Thermistor_i + rand(-2, 2) \mid \forall i \subset \{1, 2, 3\} \tag{4.2}$$

4.3. Integración

Durante el proceso de integración se unificaron todos los módulos del OBSW para ejecutar el sistema con toda la funcionalidad diseñada. En este proceso se realizaron las siguientes tareas:

Inclusión de nuevos paquetes. Se incluye la definición de paquetes de diferentes módulos (planificación y comunicaciones) que son esenciales para el correcto funcionamiento de los módulos expuestos anteriormente.

Recepción de paquetes. Se modifican las sentencias de recepción de mensajes, obligando a que cada módulo se encuentre bloqueado hasta que se reciba el mensaje de activación del planificador. La recepción de comandos del módulo de comunicaciones en la tarea de telemetría se modifica para que no sea bloqueante, por ello, se establece un tiempo de espera ínfimo que no afecte al rendimiento de este módulo.

Compilación. Se modifican las reglas de compilación y ejecución del sistema para generar el binario con todos los módulos del OBSW, véase el Apéndice C.

Capítulo 5

Pruebas y evaluación

En este capítulo se incluyen las pruebas que se han realizado que verifican el correcto funcionamiento del sistema. Además, se realiza una evaluación de rendimiento para asegurar los plazos de tiempo, ya que, es un sistema en tiempo real crítico.

5.1. Pruebas

Esta sección contiene las pruebas realizadas que validan el comportamiento del sistema según los requisitos software definidos en la Sección 3.4.

Entorno de prueba

A continuación, se describe el entorno (hardware/software) sobre el que se han ejecutado las pruebas:

- GNU/Linux (Kernel 4.18) - Ubuntu 18.04 LTS.
- Intel Core i7 4720-HQ 2.60 GHz.
- 12 GB RAM DDR3.
- SSD Samsung 860 Evo 500 GB.

En esta fase se verifica el comportamiento del sistema diseñado realizando una verificación *software-in-the-loop* [1].

Casos de prueba

En esta sección se describen los casos de prueba que se han utilizado para verificar el funcionamiento de la solución propuesta.

[1] Se verifica el funcionamiento de la solución utilizando un emulador hardware

Los casos de prueba se especifican utilizando la plantilla de la Figura 5.1. Los atributos de esta plantilla son los siguientes:

Identificador. Identifica unívocamente al caso de prueba, será CP-XX, donde XX es el número de secuencia que comienza en 01.

Descripción. Especificación del caso de prueba.

Entrada. Datos necesarios para ejecutar la prueba.

Salida. Salida esperada de la prueba.

Origen. Referencia a los requisitos software que dieron lugar a la prueba.

Entorno. Requisitos del entorno sobre el que se ejecuta la prueba.

Identificador

Descripción:

Entrada:

Salida:

Origen:

Entorno: n/a

Figura 5.1: Plantilla de casos de prueba

CP-01

Descripción:	Se desbloquea la tarea de telemetría, se leen todos los valores de monitorización, se genera el paquete y se envía al Software Bus.
Entrada:	Paquete de activación del planificador
Salida:	Se ha enviado el paquete periódico y se comprueba que no han existido errores al enviar
Origen:	F-SR-01,F-SR-05,F-SR-07,F-SR-11,F-SR-12,F-SR-15
Entorno:	n/a

CP-02

Descripción:	Se recibe el comando de petición de paquete posicional, se leen los valores de posición del nanosatélite, se genera el paquete y se envía al Software Bus.
Entrada:	Paquete de activación del planificador y comando de petición del módulo de comunicaciones
Salida:	Se ha enviado el paquete posicional y se comprueba que no han existido errores al enviar
Origen:	F-SR-02,F-SR-05,F-SR-06,F-SR-07,F-SR-11,F-SR-12,F-SR-15
Entorno:	n/a

CP-03

Descripción:	Se recibe el comando de petición de paquete térmico, se leen los valores de temperatura del nanosatélite, se genera el paquete y se envía al Software Bus.
Entrada:	Paquete de activación del planificador y comando de petición del módulo de comunicaciones
Salida:	Se ha enviado el paquete térmico y se comprueba que no han existido errores al enviar
Origen:	F-SR-03,F-SR-05,F-SR-06,F-SR-07,F-SR-11,F-SR-12,F-SR-15
Entorno:	n/a

CP-04

Descripción:	Se recibe el comando de petición de paquete solar, se leen los valores de posición del nanosatélite respecto al Sol, se genera el paquete y se envía al Software Bus.
Entrada:	Paquete de activación del planificador y comando de petición del módulo de comunicaciones
Salida:	Se ha enviado el paquete solar y se comprueba que no han existido errores al enviar
Origen:	F-SR-04,F-SR-05,F-SR-06,F-SR-07,F-SR-11,F-SR-12,F-SR-15
Entorno:	n/a

CP-05

Descripción:	Se desbloquea el control térmico que solicita la temperatura del sistema a los termistores. Se comprueba que la lectura está dentro de rango y se almacena en la tabla de telemetría.
Entrada:	Paquete de activación del planificador
Salida:	Actualizar información térmica en la tabla de telemetría
Origen:	F-SR-05,F-SR-07,F-SR-08,F-SR-13,F-SR-15
Entorno:	n/a

CP-06

Descripción:	Se desbloquea el control térmico que solicita la temperatura del sistema a los termistores. Se comprueba que la lectura está dentro de rango. Se activan los calentadores, ya que, la temperatura es inferior al umbral mínimo. Se almacena la temperatura y el estado de los calentadores en la tabla de telemetría.
Entrada:	Paquete de activación del planificador
Salida:	Actualizar información térmica en la tabla de telemetría y activar calentadores
Origen:	F-SR-05,F-SR-07,F-SR-08,F-SR-09,F-SR-13,F-SR-14,F-SR-15
Entorno:	n/a

CP-07

Descripción:	Se desbloquea el control térmico que solicita la temperatura del sistema a los termistores. Se comprueba que la lectura está dentro de rango. Se desactivan los calentadores, ya que, la temperatura es superior al umbral máximo. Se almacena la temperatura y el estado de los calentadores en la tabla de telemetría.
Entrada:	Paquete de activación del planificador
Salida:	Actualizar información térmica en la tabla de telemetría y desactivar calentadores
Origen:	F-SR-05,F-SR-07,F-SR-08,F-SR-10,F-SR-13,F-SR-14,F-SR-15
Entorno:	n/a

Matriz de trazabilidad F-SR/CP

	F-SR-01	F-SR-02	F-SR-03	F-SR-04	F-SR-05	F-SR-06	F-SR-07	F-SR-08	F-SR-09	F-SR-10	F-SR-11	F-SR-12	F-SR-13	F-SR-14	F-SR-15
CP-01	•				•		•				•	•			•
CP-02		•			•	•	•				•	•			•
CP-03			•		•	•	•				•	•			•
CP-04				•	•	•	•				•	•			•
CP-05					•		•	•					•		•
CP-06					•		•	•	•				•	•	•
CP-07					•		•	•		•			•	•	•

Figura 5.2: Matriz de trazabilidad casos de prueba - requisitos funcionales

5.2. Evaluación

En esta sección se evalúa el rendimiento del sistema diseñado. Al tratarse de un sistema en tiempo real crítico, el tiempo de cómputo de cada componente tiene que ser menor o igual que su periodo, puesto que, si es mayor se pone en riesgo la misión. Los tiempos de cómputo máximo y periodo [2] de cada tarea se encuentran en la Tabla 5.1.

Tarea	Modo de ejecución	Periodo	Cómputo máximo
Telemetría	Nominal	0.2 Hz	31.25 ms
Telemetría	Seguro	0.2 Hz	50 ms
Control térmico	Nominal	0.1 Hz	31.25 ms
Control térmico	Seguro	0.1 Hz	50 ms

Tabla 5.1: Tiempos de cómputo máximo y periodo

Para estimar el tiempo de cómputo de los módulos desarrollados se ha calculado el intervalo de confianza con una población de 30 muestras. Para calcular el intervalo de confianza se utiliza la Ecuación 5.1.

$$\mu = \bar{x} \pm z\frac{\sigma}{\sqrt{n}} \tag{5.1}$$

μ. Cota superior e inferior del intervalo.

$\bar{x}$. Media de la población.

z. Valor asociado de distribución normal.

σ. Desviación estándar.

n. Número de muestras.

Se calcula el intervalo con una confianza del 95 % para ambas tareas, los datos obtenidos se encuentran en la Tabla 5.2. Todas las mediciones están en milisegundos.

Tarea	$\bar{x}$	σ	Cota inferior	Cota superior
Telemetría	0,01770	0,0026	0,0168	0,0186
Control térmico	0,04333	0,0758	0,0162	0,0705

Tabla 5.2: Intervalo de confianza

Como se puede comprobar, ambos módulos se ejecutan significativamente más rápido que el tiempo de cómputo máximo asignado. Estas mediciones son una aproximación, ya que, se están realizando en un hardware muy superior al de la misión, puesto que, la integración y ejecución sobre el hardware definitivo corresponde a otro Trabajo de Fin de Grado. Sin embargo, el hardware sobre el que se ejecutará no es del orden de 100 a 1000 veces inferior, por ello, se puede concluir que ambos módulos cumplen con las restricciones de tiempo definidas.

[2] El tiempo de cómputo máximo y periodo han sido definidos por el planificador

Capítulo 6

Plan de proyecto

Este capítulo recoge la planificación temporal del proyecto. En la Sección 6.1 se detallan las tareas en las que se ha dividido en proyecto. Seguidamente, en la Sección 6.2 se incluye un diagrama de Gantt con la planificación temporal del mismo. En la Sección 6.3 se incluye el presupuesto del proyecto, con el cálculo de costes y amortizaciones. Finalmente, en la Sección 6.4 se detalla el entorno socio-económico, así como, el impacto del proyecto.

6.1. Tareas

El proyecto se ha dividido en diferentes tareas, son las siguientes:

Presentación del proyecto. Presentación de la Cátedra UC3M-SENER por D. Jesús Carretero Pérez, D. Javier Fernández Muñoz y D. Ignacio Melgar Bautista.

Inicio del proyecto. Asignación de Trabajos de Fin de Grado a cada uno de los integrantes de la Cátedra.

Planificación. Esta tarea se basa en la lectura de documentación y referencias útiles para el desarrollo del proyecto. Consta de las siguientes subtareas:

- **Lectura de documentación.** Lectura de referencias bibliográficas y estándares útiles para el desarrollo del sistema.
- **Estudio cFE.** Lectura de documentación de cFE, así como, de su propia API. Análisis de módulos de ejemplo predefinidos en el sistema.
- **Herramientas alternativas.** Trabajo de investigación sobre OBSW similares a cFE.

Análisis y diseño. Análisis de viabilidad y definición de requisitos. Esta tarea se compone de las siguientes subtareas:

- **Definición de requisitos.** Definición de requisitos de usuario, requisitos del sistema y casos de uso.
- **Diseño del sistema.** Definición de la arquitectura del sistema de la misión y del Trabajo de Fin de Grado.

Implementación. Desarrollo e implementación de los diferentes módulos funcionales que conforman el proyecto, son los siguientes:

Simulador térmico. Desarrollo de sensores y actuadores térmicos para simular componentes hardware. Desarrollo del módulo de control térmico.

Telemetría. Desarrollo del módulo funcional de telemetría. Obtención de valor del sensor, definición de paquete por sensor, envío de paquetes por *software bus*.

Integración. Integración de los módulos descritos previamente en el sistema global.

Pruebas y evaluación. Definición e implementación de pruebas que verifican el correcto comportamiento del sistema. Esta tarea se divide en las siguientes subtareas:

Pruebas. Pruebas que comprueban la funcionalidad del sistema.

Evaluación. Evaluación de rendimiento del sistema y verificación de plazos establecidos.

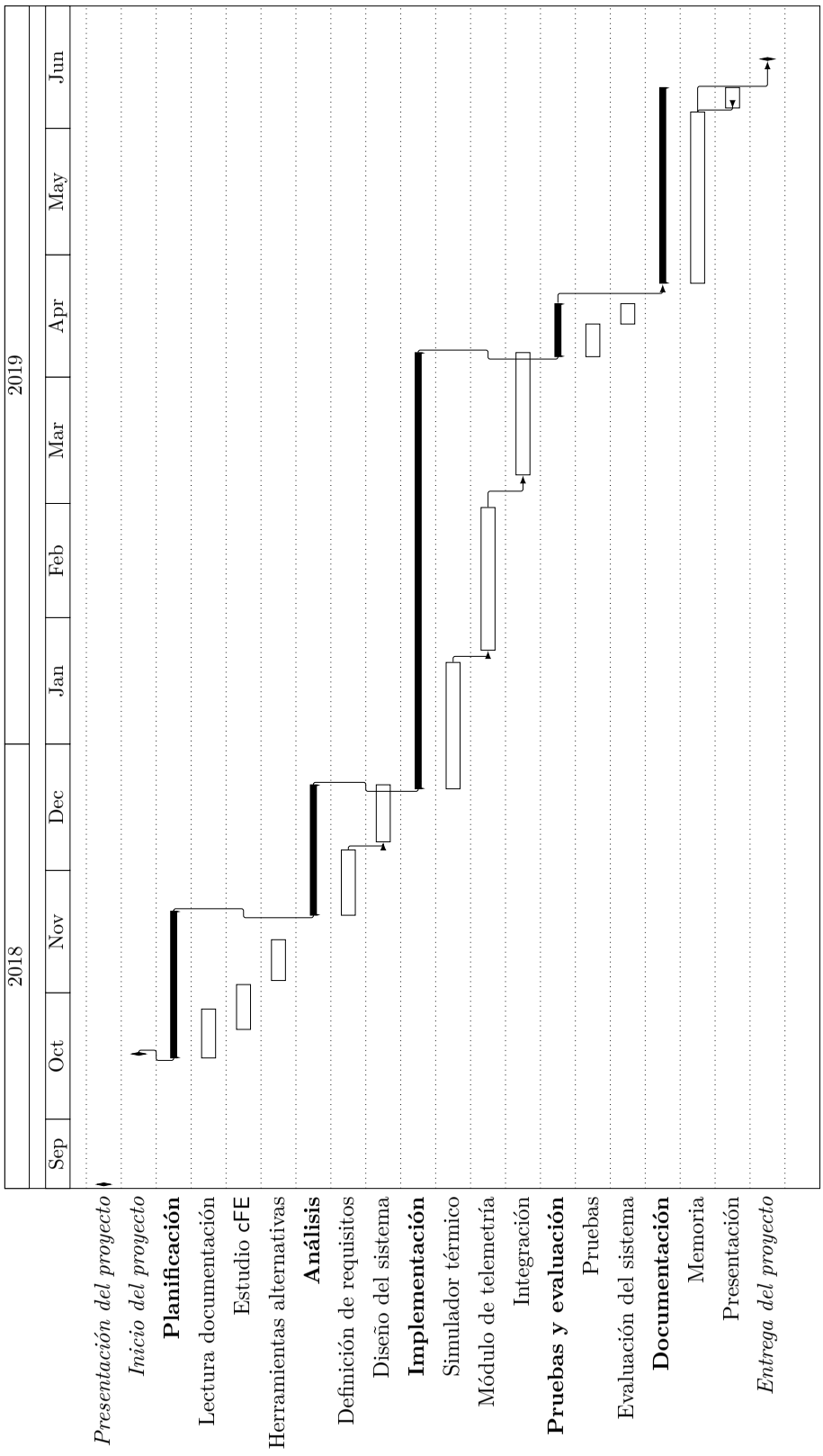
Documentación. Desarrollo del presente documento y de la presentación.

6.2. Planificación

A partir de la división de tareas definidas en la Sección 6.1, se ha diseñado la planificación temporal del proyecto mediante un diagrama de Gantt.

El diagrama de Gantt es una herramienta que se utiliza para planificar y programar las tareas de un proyecto durante un periodo de tiempo determinado. En el eje vertical quedan definidas las tareas que forman el proyecto, mientas que, en el eje horizontal se establece el periodo de duración de las mismas.

El proyecto ha tenido una duración total de 8 meses, dedicándole mensualmente una media de 50 horas. Si se descuentan días de descanso/vacaciones o inconvenientes, se obtiene una media mensual de 45 horas. Por tanto, el total de horas dedicadas al Trabajo de Fin de Grado es de 360 horas.


2018
2019
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Presentación del proyecto
Inicio del proyecto
Planificación
Lectura documentación
Estudio cFE
Herramientas alternativas
Análisis
Definición de requisitos
Diseño del sistema
Implementación
Simulador térmico
Módulo de telemetría
Integración
Pruebas y evaluación
Pruebas
Evaluación del sistema
Documentación
Memoria
Presentación
Entrega del proyecto

6.3. Presupuesto

En esta sección se detalla el presupuesto, en base a la planificación y duración del proyecto expuesto en la Sección 6.2. Los costes se dividirán en dos bloques: costes directos (costes asociados al personal y al material empleado) y costes indirectos (costes que influyen indirectamente al proyecto). Los costes no incorporan el 21 % de I.V.A., que será incluido en el resumen de costes, en la Tabla 6.7.

Costes directos

Los costes directos son aquellos que están relacionados directamente con la realización y construcción de los productos o servicios que ofrece una empresa. Los costes directos se pueden dividir en dos bloques, son los siguientes:

Coste del personal. El coste del personal varía en función de su titulación, experiencia laboral y situación geográfica.

Costes de material. Son todos aquellos gastos generados por la compra de herramientas para desarrollar el proyecto. Se pueden dividir en material hardware y material software.

El proyecto se ha dividido en diferentes tareas que fueron descritas en la Sección 6.1. En la Tabla 6.1 se puede comprobar la cantidad de horas que ha requerido cada tarea.

Tarea	Horas
Análisis y diseño	70 h
Implementación	90 h
Pruebas	30 h
Documentación	170 h
Total	360 h

Tabla 6.1: División de tareas

Los costes asociados al personal se calculan utilizando la Ecuación 6.1.

$$Coste = Horas\ totales\ *\ coste\ por\ hora \tag{6.1}$$

Los costes del personal se encuentran detallados en la Tabla 6.2.

Cargo	Horas	Coste/Hora	Total
Analista	40 h	45.00 €	1800.00 €
Analista/Programador	120 h	28.00 €	3360.00 €
Programador	200 h	17.00 €	3400.00 €
Total	360 h	8560.00 €	

Tabla 6.2: Costes de personal

Los costes asociados al material hardware hacen referencia a todos los equipos y componentes empleados en el proyecto. Los costes del material hardware están detallados en la Tabla 6.3. Además, en la Tabla 6.5 se puede ver el coste amortizado de estos productos.

Producto	Precio
Ordenador portátil	797,48 €
SSD Samsung EVO 860 (500 GB)	66.36 €
Ratón Logitech M185	8.61 €
Monitor externo	118.50 €
Cable HDMI	2.71 €
Total	**993.66 €**

Tabla 6.3: Costes de material hardware

Los costes asociados al material software hacen referencia a todas las herramientas informáticas empleadas en el proyecto. Los costes del material software se recogen en la Tabla 6.4.

Producto	Precio
Ubuntu 18.04 LTS	0 €
Licencia educativa anual CLion	0 €
Licencia educativa Microsoft Office	0 €
Visual Paradigm Community Edition	0 €
MiKTeX	0 €
Texmaker	0 €
Git	0 €
Total	**0 €**

Tabla 6.4: Costes de material software

El material hardware y software empleado en el proyecto pierde su valor anualmente debido a su desgaste, esto se denomina amortización. Aunque hay diferentes tipos de amortización, en este proyecto se utilizará la amortización lineal, ya que, se supone que el desgaste anual será siempre el mismo.

La amortización lineal se calcula utilizando la Ecuación 6.2.

$$A = \frac{X}{Y} * Z * R \tag{6.2}$$

A. Amortización lineal que se calcula.

X. Tiempo de utilización del material, es decir, la duración del proyecto.

Y. Vida útil. Generalmente la vida útil del material es de cinco años.

Z. Coste del material especificado en la Tabla 6.3 y en la Tabla 6.4.

R. Coeficiente. Es el porcentaje de uso que se dedica al proyecto, en este caso el 100 %.

La amortización calculada está recogida en la Tabla 6.5. La duración del proyecto y la vida útil se mide en meses.

Producto	Coste	Coeficiente (%)	Duración del proyecto	Vida útil	Amortización
Portátil	797.48 €	100	8	60	106.33 €
SSD	66.36 €	100	8	60	8.49 €
Ratón	8.61 €	100	8	60	1.15 €
Monitor	118.50 €	100	8	60	15.80 €
Cable HDMI	2.71 €	100	8	60	0.36 €
Total					**132.13 €**

Tabla 6.5: Amortización de productos

Costes indirectos

Los costes indirectos están presentes durante el proceso de producción, pero no se pueden asignar a cada uno de los productos de forma directa.

Para el gasto energético se ha utilizado un coste de 0.14 €/kWh [32]. El gasto máximo del portátil es de 120 Wh, se supone un consumo medio de 80 Wh. Además, se computa el coste de una línea de fibra óptica simétrica de 100 Mbps [33].

Asimismo, se deben imputar los costes del material de oficina empleado, que engloba folios, cartuchos de impresora, bolígrafos, etc. También, se computan los costes asociados al desplazamiento.

Los costes indirectos se recogen en la Tabla 6.6.

Recurso	Coste
Electricidad	3.58 €
Fibra simétrica	102.72 €
Material de oficina	15.80 €
Transporte	160 €
Total	**282.10 €**

Tabla 6.6: Costes indirectos

Resumen de costes

A partir de los datos obtenidos en las secciones anteriores se calcula el precio de venta del proyecto. En la Tabla 6.7 se incluye un resumen del presupuesto.

Para subsanar diferentes imprevistos (baja médica del personal, fallos o roturas en el material, etc.) se añade un margen del 10 % sobre el coste. Además, se aplica un margen de beneficio del 25 % sobre el precio, incluyendo imprevistos, para obtener rentabilidad del proyecto.

Descripción	Coste
Costes de personal	8560.00 €
Costes de material	0 €
Amortización	132.13 €
Costes indirectos	282.10 €
Margen de imprevistos (10 %)	897.46 €
Margen de beneficio (25 %)	2468.02 €
Total sin I.V.A.	**12340.06 €**
Total	**14931.47 €**

Tabla 6.7: Resumen de costes

Por tanto, el precio final de este proyecto asciende a la cantidad de **14 931.47 €(CATORCE MIL NOVECIENTOS TREINTA Y UN EUROS Y CUARENTA Y SIETE CÉNTIMOS)**.

6.4. Entorno socio-económico

Como se ha comentado en la Sección 2.1, en 1957 la URSS envía el primer satélite artificial realizado por el ser humano al espacio. EEUU contraataca lanzando su primer satélite en 1958 y da comienzo a la carrera espacial. A partir de ese momento, la inversión económica en esta disciplina aumenta de manera exponencial. En apenas una década, el ser humano consigue llegar a la Luna.

A día de hoy, es imposible vivir sin los avances e inventos que han surgido como consecuencia de la industria espacial (satélites de comunicación, geolocalización, etc.). Estos avances facilitan la vida de la humanidad en gran medida. A continuación, se analizan los principales factores que afectan a esta industria.

Factores tecnológicos

La conquista espacial ha provocado la aparición de múltiples artilugios que son fundamentales a día de hoy. Además, se han comprobado teorías científicas y han aparecido nuevas que podrían revolucionar la industria espacial en el futuro.

Algunos de las tecnologías más importantes que aparecen como consecuencia de esta industria son las siguientes:

Geolocalización (GPS). Con al menos tres satélites artificiales (órbitas geoestacionarias) es posible conocer la posición de un dispositivo con gran precisión. Estos sistemas se utilizan en sistemas de navegación, tareas de rescate, etc.

Comunicación. La comunicación con dispositivos móviles está establecida en la sociedad. Estos dispositivos se comunican mediante redes móviles, sin embargo, en determinados lugares del planeta (océanos, polos, etc) no existe cobertura. Por ello, en estos lugares el dispositivo móvil tiene acceso a la red móvil conectándose a los satélites artificiales.

Energías. Los satélites artificiales no funcionan con combustibles fósiles cuando se encuentran en órbita. Para que funcionen los sistemas se aprovecha la energía solar mediante placas solares. Estas placas se han comercializado y se utilizan en elementos de la vida cotidiana (generadores, dispositivos de señalización,etc).

Factores políticos

En la actualidad, existen gran cantidad de agencias espaciales, siendo las más importantes la NASA y la ESA. Algunos países cuentan con su propia agencia espacial, sin embargo, es común que se integren en otra agencia mayor.

Al igual que muchos países de Europa, España no dispone de agencia espacial propia, por ello, estos países son miembros de la ESA.

Factores económicos

En las guerras la inversión en ciencia y tecnología que realizan los países aumenta considerablemente para ganar el conflicto, esto ocurrió también en la carrera espacial. EEUU y la URSS invirtieron mucho dinero para convertirse en la primera potencia a nivel mundial.

En la actualidad, muchas agencias espaciales colaboran e intercambian información para lograr nuevos avances. Se espera que en la próxima década el ser humano vuelva a la Luna.

Además, existen multitud de compañías privadas que invierten grandes cantidades económicas para conseguir nuevos avances tecnológicos. Una de estas compañías privadas ofrece viajes por el espacio a pasajeros privados.

Impacto socio-económico

La industria espacial ha crecido significativamente durante el último medio siglo. Inicialmente, las misiones espaciales eran realizadas exclusivamente por agencias gubernamentales, sin embargo, cada vez más compañías privadas forman parte de este sector. Por tanto, aumenta el número de trabajadores en la sociedad y el avance tecnológico es mayor.

Como parte de la Cátedra Universidad Carlos III - SENER, se espera que el proyecto tenga un gran impacto en la industria espacial, desde el punto de vista académico/universitario, sentando las bases para las futuras iteraciones de esta colaboración. Se conseguirá que el nanosatélite sea capaz de tomar fotografías de la Tierra que se utilizarán con propósitos científicos y académicos, además de estudiar los efectos de diferentes tipos de radiación sobre los componentes del sistema.

Capítulo 7

Marco regulador

En el presente capítulo se detallan las principales leyes y estándares que afectan/condicionan a este proyecto.

7.1. Legislación y normativa internacional

Tratado sobre el espacio exterior

El Tratado sobre el espacio exterior [34], cuyo nombre completo es "Tratados y principios de las Naciones Unidas sobre el espacio ultraterrestre", es un tratado sobre el que se fundamenta el Derecho internacional sobre el espacio. Inicialmente, fue aprobado por Estados Unidos, Reino Unido y la Unión Soviética en 1967, en la actualidad, ha sido firmado por más de 105 países.

Las principales características [35] de este tratado son las siguientes:

- Principios que deben controlar las actividades de los Estados en la exploración del espacio ultraterrestre, incluso la Luna y otros cuerpos celestes.
- Acuerdo sobre el salvamento y devolución de astronautas.
- Responsabilidad internacional por daños causados por objetos espaciales.
- Registro de objetos lanzados al espacio ultraterrestre.

CCSDS

El Consultative Committee for Space Data Systems [36] fue fundado en 1982 para que las agencias espaciales de los países miembros participasen en el desarrollo de estándares de comunicaciones y datos espaciales. Actualmente, colaboran multitud de expertos de los 27 países miembros en el desarrollo de estándares.

El objetivo principal de esta organización es conseguir que todas las misiones espaciales (gubernamentales y comerciales) sigan su normativa para reducir riesgos, tiempo de desarrollo y costes. En la actualidad, más de 900 misiones han seguido estos estándares.

ECSS

La European Cooperation for Space Standardization [37] es un organismo que se fundó para desarrollar un conjunto de normativas espaciales comunes a la Unión Europea. En esta organización participan las principales agencias europeas, son las siguientes:

- European Space Agency (ESA).
- Agenzia Spaziale Italiana (ASI).
- UK Space Agency.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR).

Como se explicó en la Sección 3.1, se sigue el estándar PUS (ECSS-E-ST-70-41C) de la ESA.

7.2. Legislación y normativa nacional

CNAF

El Cuadro Nacional de Atribución de Frecuencias [38] es el encargado del control de radiofrecuencias en el territorio nacional. Determina el uso de las frecuencias que van desde los 8.3 kHz hasta los 3000 GHz.

La primera versión de este documento se publicó en 1990. Es un documento que requiere actualizaciones constantes debido a la normativa de los organismos internacionales (UIT, CEPT, UE y ETSI) con competencias regulatorias a los que pertenece España.

El CNAF se organiza en cuatro columnas, son las siguientes:

Primera. Atribución de las bandas de frecuencias según la UIT.

Segunda. Atribución nacional de cada banda de frecuencias, habitualmente coincide con la primera columna.

Tercera. Tipo de uso del espectro, se divide en las siguientes modalidades:

- **C.** Uso común.
- **E.** Uso especial.
- **P.** Uso privativo.
- **R.** Uso reservado al Estado.
- **M.** Uso mixto entre P y R.

Cuarta. Observaciones del Reglamento de Radiocomunicaciones y Notas de utilización Nacional (NN).

Capítulo 8

Conclusiones y trabajos futuros

En este capítulo se detallan las conclusiones obtenidas después de realizar el presente Trabajo de Fin de Grado. En primer lugar, en la Sección 8.1 se verifica que se han cumplido los objetivos que se establecieron en la Sección 1.3. Seguidamente, en la Sección 8.2 se exponen las conclusiones del proyecto. En la Sección 8.3 se exponen las conclusiones personales del proyecto. Finalmente, en la Sección 8.4 se exponen una serie de tareas que se podrían realizar en el futuro para completar la misión.

8.1. Objetivos

A continuación, se verifica la consecución de objetivos:

- Se ha participado en el diseño de la arquitectura global del sistema, consiguiendo un sistema modular en el que todos los componentes están comunicados entre ellos. Todos los componentes son independientes, por ello, el mantenimiento de los mismos y la posibilidad de añadir nueva funcionalidad en el futuro es viable.
- Se ha desarrollado un simulador térmico totalmente funcional. Se simula a nivel software lecturas de la temperatura del sistema y un aumento en las mismas después de activar los calentadores. El control de estos componentes (sensores y actuadores) es realizado por el control térmico, que es un módulo del OBSW.
- Se ha desarrollado el módulo de telemetría que permite realizar una monitorización completa del sistema. Se implementan los servicios que ofrece cFE para conseguir comunicación con otros módulos del OBSW (planificador y comunicaciones) y se utiliza el sistema de eventos para realizar un *log* del módulo. Este módulo es capaz de generar un paquete periódico con toda la información disponible del sistema, además, recibe comandos desde el segmento terrestre, los interpreta y envía la información solicitada.
- Como extensión del módulo de telemetría, se ha desarrollado otro módulo adicional, la tabla de telemetría, que almacena los valores de monitorización del sistema y es accesible por todos los módulos del OBSW que necesiten consultar o actualizar esta información.

8.2. Proyecto

El Trabajo de Fin de Grado ha necesitado un alto grado de implicación, compromiso y coordinación por ser parte de un proyecto más grande. El proyecto ha requerido reuniones semanales con los responsables del resto de componentes y con el cliente, SENER, para fijar objetivos, obtener requisitos y verificar resultados. Los primeros meses fueron esenciales para realizar estas tareas, por ello, la implementación del sistema no comenzó hasta casi el presente año.

Se ha comprendido la arquitectura que se utiliza en este tipo de misiones. Debido a las estrictas restricciones de tiempo, se utiliza un sistema operativo en tiempo real, RTOS, sobre el hardware para garantizar los plazos de ejecución. Se utiliza una capa de abstracción del sistema operativo, OSAL, que abstrae al sistema del OS/ RTOS (Linux, RTEMS, etc) sobre el que se ejecuta. Finalmente, en la última capa se encuentran los módulos desarrollados por el usuario que hacen uso de los servicios de cFE.

La fase de implementación e integración duró más de lo esperado, ya que, se tuvieron problemas que no se contemplaron en un principio. Se han tenido muchos problemas con la compilación del sistema, puesto que, no se ha encontrado nada de documentación en Internet. Sin embargo, una vez se comprendió el funcionamiento de cFE, implementar sus servicios en las aplicaciones no resultó muy complicado, ya que, su código incluye documentación generada con Doxygen.

8.3. Personales

El Trabajo de Fin de Grado ha sido unos de mis mayores retos a nivel personal hasta la fecha necesitando el máximo grado de implicación. La Cátedra es una oportunidad fantástica para los estudiantes que les guste el espacio, ya que, tendrán la oportunidad de participar en un proyecto educativo que algún día se convertirá en realidad.

Se ha profundizado en los conocimientos adquiridos en la asignatura “Sistemas en tiempo real”, fundamentales para comprender y diseñar el sistema. Los conocimientos adquiridos en la rama de Ingeniería de Software han sido esenciales para el desarrollo del Capítulo 3. Se ha utilizado el paquete SRS [39] de Javier López que automatiza el proceso de generación de tablas de requisitos, casos de uso, etc.

Se ha aprendido LaTeXpara realizar el presente documento, *TikZ* para la generación de diversas figuras y *pgfgantt* para el diagrama de Gannt del proyecto.

8.4. Trabajo futuro

Al ser el primer año de la Cátedra Universidad Carlos III - SENER, el proyecto aún se encuentra en una fase muy inicial. En el futuro, se debe seguir ampliando la funcionalidad del sistema para lograr el objetivo final del proyecto global. Este proceso se debe realizar iterativamente y en cada iteración se debe añadir nueva funcionalidad, además de actualizar los componentes previos.

A continuación, se detallan algunas de las tareas que se podrían realizar en las siguiente iteraciones del proyecto:

Integración con Simulink. Se deberían integrar los modelos realizados en Simulink por otros compañeros de diferentes departamentos. De esta forma se consiguen añadir sistemas que logran una simulación realista fundamentada en modelos matemáticos/físicos reales (*model-in-the-loop*).

Módulo de gestión de memoria. Este módulo debe permitir la carga de nuevos *payloads* procedentes del segmento terrestre. De esta forma, se pueden definir nuevos paquetes, actualizar el sistema para corregir errores, añadir nueva funcionalidad, etc.

Generador de documentación. Para facilitar el trabajo en siguientes iteraciones, se podría utilizar Doxygen, que genera documentación de manera automática en función del código desarrollado.

Incluir nuevos sensores/actuadores. La misión no está completa con los sensores y actuadores que se han utilizado en esta primera iteración, por ello, el cliente debe definir todos aquellos componentes que se utilizan en una misión real.

Apéndice A

Summary

A.1. Introduction

Control systems are responsible for controlling the behavior of a given physical environment. These systems always perform the same algorithm, iteratively. First of all, information is received from the physical environment by means of sensors. The received information is analysed and a decision is made on what should be modified in the physical environment to achieve the desired behaviour. Finally, if needed, appropriate changes are made to the physical environment using actuators.

The control system to be developed is a real-time critical system based on a RTOS. For a control system to be critical, it must meet two conditions. The first one is that the system must execute the correct actions. The second one is that the system performs the actions on schedule. Failure to comply with one of these two conditions results in total system failure. If the system fails, the consequences are very serious, both economically and humanly. This type of system is very common in different fields such as aviation, railway systems, car control, etc.

The purpose of the Cathedra Universidad Carlos III and SENER, is to develop a nanosatellite, specifically a university *CubeSat*, that works using a RTOS. The mission of this *CubeSat* is to take high-resolution pictures of the Earth that will be used for scientific purposes.

The CubeSat standard was defined jointly in 1999 between the California Polytechnic University and Stanford University, with the aim of developing small spacecraft in university classrooms.

The task assigned to the Degree in Computer Engineering, consists in the development and implementation of the OBSW, which will control the nanosatellite in the space segment, and the development and implementation of the Ground System, which will control the mission from the ground segment. Both segments have to be in constant communication to check that the system operates correctly, therefore, there must be a data link, see Figure A.1.

The OBSW consists of different modules that make the system operate correctly at software and hardware level. The task assigned to this Bachelor's thesis is the design and implementation of the telemetry module and the thermal simulator.

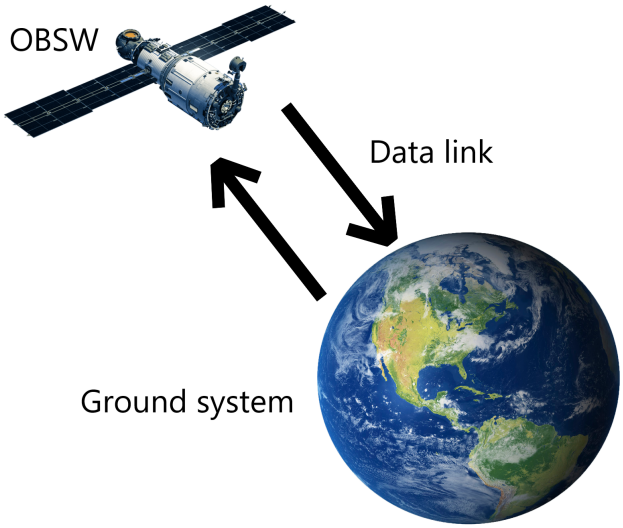


Figura A.1: Data link between OBSW and Ground System

Problem statement

The bad design and implementation of the different modules that make up the OBSW, imply the total failure of the mission. Therefore, the system architecture and each of the components that make up the OBSW must be known. Also, the components that make up the Ground System must be well designed and implemented to observe the state of the nanosatellite from the ground segment and modify its behavior, if necessary.

Goals

The objectives of this project are detailed below:

Definition of the system architecture. To design the system architecture with the partners of the OBSW.

Design and implementation of the thermal simulator. Definition, design and implementation of the different sensors and thermal actuators that make up the system. These components and the thermal control will be simulated at software level.

Design and implementation of the telemetry module. Definition and implementation of the telemetry module. Packages must be defined and implemented. In addition, communication with the modules of the OBSW must be achieved using Software Bus.

A.2. cFE

Core Flight Executive, cFE, is a development application and execution environment developed by the U.S. National Aeronautics and Space Administration, NASA.

This environment provides a set of services that facilitate the development and implementation of modules for space missions. It incorporates the following services:

Software Bus This service is responsible for establishing communication between all components of the system.

Time This service implements a control system to manage the start and end of tasks and notify to the ground segment.

Events This service combined with Time service allows you to have a total record of all the actions that have been executed by the system.

Executive This service initiates and executes the environment.

Table services This service allows you to define the tasks to be executed by the system, their priority, etc. That is, it is the system scheduler.

cFE provides developers with the application programming interface, API, for each of the services it implements. In addition, it includes a number of development tools that are essential in these missions, are as follows:

- A unitary test framework that allows users to check that developed applications are working correctly.
- Time Analyzer provides real-time performance. This tool is essential, as it allows to know the behavior of the system in an embedded system before extrapolating it to the real hardware of the mission.
- Builder of task tables for the scheduler.
- Different utilities for command management and telemetry.

However, cFE is one of the components that form CFS, that is, the core of the flight system. CFS is an independent software project that is allowed to reuse, as well as, its applications.

cFE's main mission is to lay the foundations for a platform that uses reusable standalone software. In addition, along with the abstraction layer of the system, OSAL, the code is made to work on any operating system in real time (VxWorks, RTEMS, etc) without having to be modified. It is also intended to create a standard for these missions, preventing each space agency from using its own.

In conclusion, cFE's purpose is to develop software for embedded systems that can be tested on personal computers without modifying the original code.

The CFS system has a modular architecture segmented in layers, see Figure A.2. The layers that compose this system are as follows:

Hardware. These are the different hardware components that compose the system.

Operating System. The operating system on which the system runs. It can be a RTOS (RTEMS, VxWorks, etc) or not (Linux).

OSAL. Layer that abstracts the system from the OS or RTOS on which the mission is executed.

cFE Services. Services offered by the cFE system.

CFS/ User Applications/Libraries. A set of example applications that includes the system and applications developed by the user.

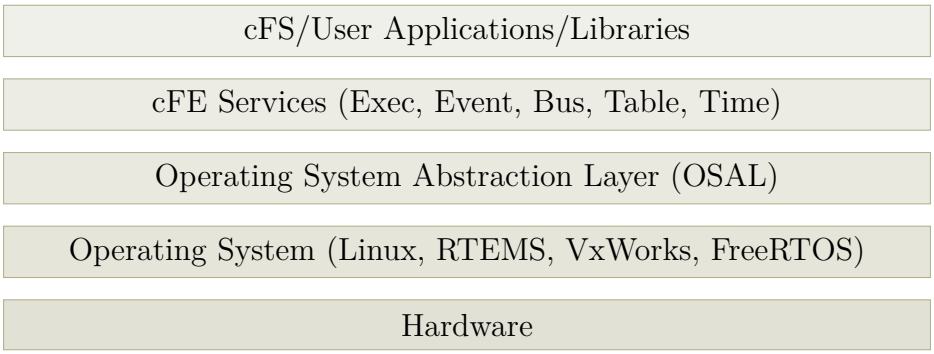


Figura A.2: CFS system architecture

A.3. System architecture

The architecture of the system, see Figure A.3, for this mission has been defined by the members of the OBSW, Ground System and the project manager. This architecture corresponds to the first design phase of the project and will be modified according to the progress of the Cathedra until the nanosatellite is placed in orbit.

The system is divided into three distinct subsystems, they are as follows:

Hardware. Hardware components (sensors and actuators) that check the environment and take action, if necessary. In this first phase of development, the hardware will be simulated, since, the components correspond to different projects of other departments.

OBSW. The OBSW is the on board software of the mission, consists of different modules that ensure that the satellite operates properly from space.

Ground System. The Ground System is the segment that is responsible for storing the data sent from the OBSW, in addition, can modify its behavior using commands.

System architecture

The components highlighted in the Figure A.3 have been implemented for this project.

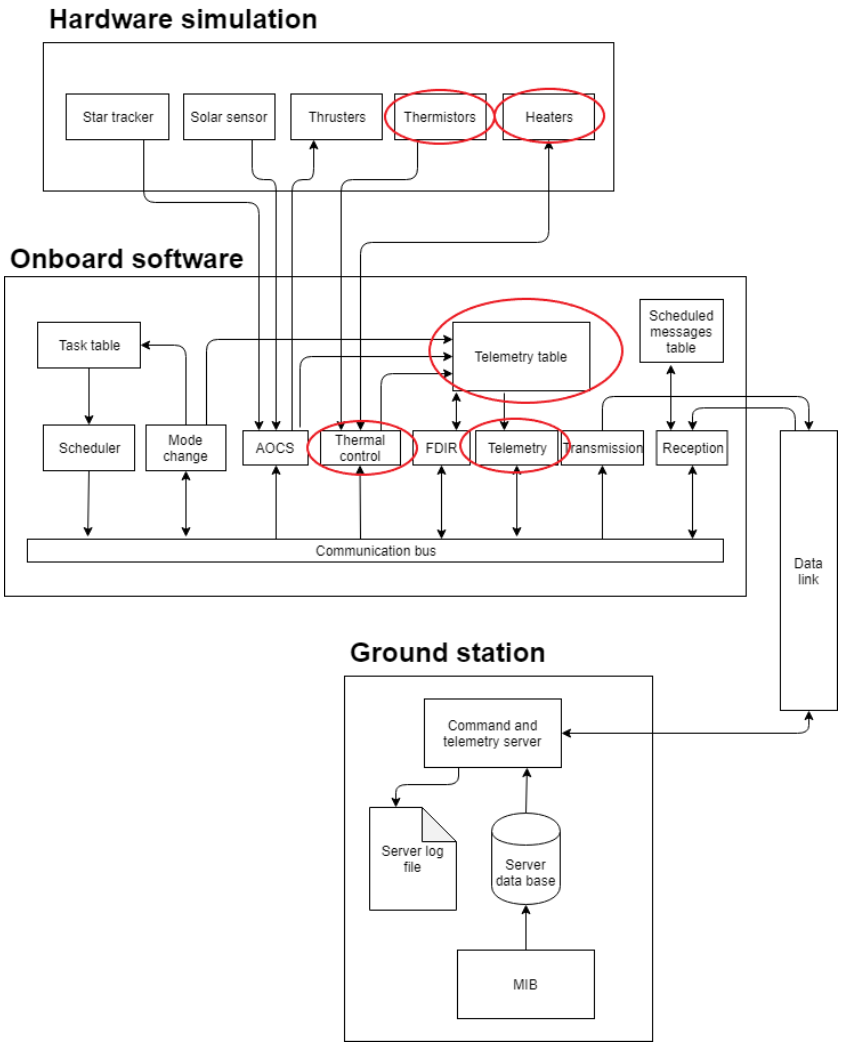


Figura A.3: System architecture

A.4. Implementation

Telemetry module

The telemetry module monitors the behavior of the system and generates the corresponding packages that the Ground System will receive. All the telemetry packages have a similar structure, they are formed by the header and the user data.

To avoid errors with floating numbers, all information will be coded in `int32_t` with an accuracy of 10^3, that is, the floating value will be multiplied by 1000 and a type transformation will be made to the type shown above. This transformation is done to avoid precision losses and because many embedded systems do not support floating arithmetic in hardware.

There are four types of packages, they are the following ones:

Periodic package. This package contains all available system information and is sent periodically.

Thermal package. This package contains all the information relating to thermal control. It is sent when the telemetry module receives a command from the communication module requesting this information.

Positional package. This package contains all the information relating to the position of the nanosatellite. It is sent when the telemetry module receives a command from the communication module requesting this information.

Solar package. This package contains all the information related to the position of the nanosatellite respect the Sun. It is sent when the telemetry module receives a command from the communication module requesting this information.

In order to optimize the size of the packages, a configuration flag will be included that allows to know the state of several components of the system. A variable of the type `int32_t` will be used where each bit corresponds to a certain component or aspect of the system.

The first 24 bits are used for sensors and actuators information. Currently, only 13 bits are used, the remaining bits are reserved for new sensors and actuators. The remaining 8 bits of the variable are used for system control. Similarly, 7 bits are reserved that can be used to control other system parameters. Bit 24 indicates the execution mode.

Thermal simulator

The thermal simulator is a software simulation of those hardware components related to the control and management of the nanosatellite temperature. It has three components, they are the following ones:

Thermistors. These sensors obtain the temperature of the system simulating a wavesine that oscillates between $233K$ and $203K$.

Heaters. These actuators modify the system temperature to avoid critical errors, being activated by thermal control.

Thermal control. This module controls the temperature of the system. It obtains the thermal data through the thermistors, verifies that they are in the defined thresholds and activates or deactivates the heaters to modify the temperature.

In order to achieve a more realistic simulation, a noise factor is introduced to the value read from the thermistors.

A.5. Project plan

Tasks and planning

The project has been divided into different tasks, they are as follows:

Project Presentation. Project Presentation by Jesús Carretero Pérez, Javier Fernández Muñoz and Ignacio Melgar Bautista.

Project start. Projects distribution of the Cathedra.

Planning. Reading of documentation and useful references for the development of the project.

Analysis and design. Analysis, definition of requirements and system design.

Implementation. Development and implementation of the different functional modules that compose this project.

Testing and evaluation. Definition and implementation of tests that verify system operation and performance evaluation.

Documents. Development of the present document and the presentation.

From the division of tasks defined above, the time schedule of the project has been designed using a Gantt chart.

The project has had a total duration of 8 months, dedicating an average of 50 hours per month. If you discount days of rest/vacation or inconveniences, you get a monthly average of 45 hours. Therefore, the total number of hours dedicated to this project is 360 hours.

Budget

The costs will be divided into two types: direct costs (costs associated with personnel and materials used) and indirect costs (costs indirectly influencing the project). The costs do not include 21 % of I.V.A., which will be included in the summary of costs.

Direct costs

Direct costs are those directly related to the realization and construction of products or services offered by a company. The personnel costs are detailed in the Table A.1.

The hardware and software material used in the project loses its value annually, this is called amortization. The calculated amortization is shown in the Table A.2. Project duration and useful life are measured in months.

Indirect costs

Indirect costs are present during the production process, they are shown in the Table A.3.

Job title	Hours	Cost/hour	Total
Analyst	40 h	45.00 €	1800.00 €
Analyst/Programmer	120 h	28.00 €	3360.00 €
Programmer	200 h	17.00 €	3400.00 €
Total	360 h	8560.00 €	

Tabla A.1: Personnel costs

Product	Cost	Ratio (%)	Project duration	Useful life	Amortization
Laptop	797.48 €	100	8	60	106.33 €
SSD	66.36 €	100	8	60	8.49 €
Mouse	8.61 €	100	8	60	1.15 €
Monitor	118.50 €	100	8	60	15.80 €
HDMI cable	2.71 €	100	8	60	0.36 €
Total					**132.13 €**

Tabla A.2: Amortisation costs

Resource	Cost
Electricity	3.58 €
Symmetrical fiber	102.72 €
Office equipment	15.80 €
Transport	160 €
Total	**282.10 €**

Tabla A.3: Indirect costs

A margin of 10 % is added to the cost in order to rectify various contingencies. In addition, a profit margin of 25 % is applied to the price, including contingencies, in order to make the project profitable. A summary of the budget is included in the Table A.4.

Description	Cost
Direct costs	8560.00 €
Indirect costs	282.10 €
Contingency margin (10 %)	897.46 €
Profit margin (25 %)	2468.02 €
Total	**14931.47 €**

Tabla A.4: Summary of costs

Therefore, the final price of this project is **14931.47 €**.

A.6. Conclusions

Goals

The objectives that have been achieved after the realization of this project are the following:

- It has participated in the design of the overall architecture of the system, achieving a modular system in which all components are communicated between them. All the components are independent, therefore, their maintenance and the possibility of adding new functionality in the future is viable.

- A fully functional thermal simulator has been developed. Thermistors simulate temperature readings and heaters simulate an increase in temperature. The control of these components (sensors and actuators) is performed by the thermal control, which is a module of the OBSW.

- The telemetry module has been developed to allow a complete monitoring of the system. cFE services are implemented to achieve communication with other modules of the OBSW (scheduler and communications module) and the event system is used to track this module. This module generates and sends a periodic package with all the available information of the system, in addition, it receives commands from the ground segment, interprets them and sends the requested information.

- As an extension of the telemetry module, another additional module has been developed, the telemetry table, which stores the system monitoring values and is accessible by all the OBSW modules that need to consult or update this information.

Project

This project has required a high degree of involvement, commitment and coordination as part of a larger project. The project has required weekly meetings with project partners and the client, SENER, to set objectives, obtain requirements and verify results. The first few months were essential to carry out these tasks.

The architecture used in this type of mission has been understood. Due to strict time constraints, a real-time operating system is used, RTOS, on the hardware to ensure execution times. It uses an abstraction layer of the operating system, OSAL, which abstracts the system from the RTOS (VxWorks, RTEMS, etc) or OS (Linux) on which it runs. Finally, in the last layer are the modules developed by the user using the cFE services.

The implementation and integration phase lasted longer than expected, as there were problems that were not contemplated in the beginning. There have been many problems with the compilation of the system, since no documentation has been found on the Internet. However, once the operation of the system was understood, implementing cFE services in the applications was not very complicated, as its code includes documentation generated with Doxygen.

Future work

As this is the first year of Cathedra Universidad Carlos III - SENER, the project is still in a very early stage. In the future, the functionality of the system must continue to be extended in order to achieve the final objective of the global project. This process must be carried out iteratively and in each iteration new functionality must be added, in addition to updating the previous components.

The main tasks that could be carried out in the future are as follows:

Integration with Simulink. Models made in Simulink by other colleagues from different departments should be integrated. In this way you can add systems that achieve a realistic simulation based on real mathematical/physical models (model-in-the-loop).

Memory module. This module must allow the loading of new payloads from the ground segment. In this way, you can define new packages, update the system to correct errors, etc.

Document generator. To facilitate the work in subsequent iterations, Doxygen could be used, which automatically generates documentation based on the code developed.

Add new sensors/actuators. The mission is not complete with the sensors and actuators that have been used in this first iteration, so the customer must define all those components that are used in a real mission.

Apéndice B

Glosario

API. Application Programming Interface, es un conjunto de funciones que permite la comunicación entre componentes software mediante abstracción.

cFE. Core Flight Executive, aplicación de desarrollo y entorno de ejecución creada por NASA.

CFS. Core Flight System, proyecto de software reutilizable creado por NASA para el desarrollo de misiones espaciales.

CPU. Central Processing Unit, componente hardware que interpreta las intrucciones de un programa informático mediante la realización de operaciones aritméticas/lógicas.

ESA. European Space Agency, organización europea dedicada a la exploración espacial.

Framework. Conjunto estandarizado de conceptos que se utiliza para resolver nuevos problemas similares.

GPS. Global Positioning System, sistema utilizado para determinar la posición de un objeto con muy alta precisión.

GUI. Graphical User Interface, es la interfaz que permite interactuar al usuario de manera sencilla con el sistema operativo.

IDE. Integrated Development Enviroment, es una aplicación informática que proporciona servicios integrales al programador para facilitar el desarrollo software.

LTS. Long Term Support, usado para el nombrado de versiones diseñadas para tener soporte durante un periodo de tiempo más largo de lo habitual.

OBSW. On-Board SoftWare, sistema software que se ejecuta en la nave de la misión espacial.

Open source. Modelo de desarrollo software basado en la colaboración abierta.

OS. Operating System, sistema software que gestiona los recursos hardware y software del computador. Además, proporciona servicios comunes para los programas del sistema.

OSAL. Operating System Abstraction Layer, capa que abstrae al sistema del hardware sobre el que se ejecuta. Implementa una interfaz genérica para servicios RTOS y una interfaz genérica para hardware.

Payload. Conjunto de datos transmitidos en un mensaje, excluyendo cabeceras o metadatos.

RTOS. Real Time Operating System, sistema operativo desarrollado para aplicaciones en tiempo real. Ejecuta las tareas en los intervalos de tiempo preestablecidos.

UDP. User Datagram Protocol, protocolo a nivel de transporte basado en el intercambio de datagramas, no es necesario haber establecido previamente una conexión.

Apéndice C

Manual de usuario

Instalación del sistema

La instalación del sistema se ha realizado en Ubuntu 18.04 LTS, en su última versión estable. Para poder construir el entorno de ejecución, ha sido necesario instalar la herramienta, CMake y el paquete `build-essentials`, que incluye los paquetes más comunes para un entorno de compilación de software, véase el Listado C.1.

```
$ sudo apt-get install cmake
$ sudo apt-get install build-essential
```

Listado C.1: Instalación herramientas previas

Una vez se han instalado las herramientas anteriores, se procede a descargar el núcleo de vuelo del repositorio de NASA en GitHub [40], para ello, se deben ejecutar una serie de comandos, véase el Listado C.2.

```
# Se clona el repositorio cFE de NASA
$ git clone https://github.com/nasa/cFE.git

# Se actualizan los modulos del sistema
$ cd cFE
$ git submodule init
$ git submodule update
```

Listado C.2: Instalación CFS

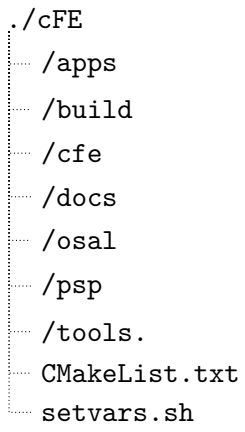


Figura C.1: Jerarquía de directorios del sistema

Jerarquía de directorios

Después de instalar CFS, que contiene los servicios de cFE, se generarán diferentes ficheros y directorios en la ruta donde se ha clonado el repositorio, véase la Figura C.1. Los directorios generados son los siguientes:

apps. Incluye las aplicaciones/librerías desarrolladas por el usuario.

build. Incluye diferentes directorios que simulan las CPUs del sistema sin necesidad de utilizar hardware.

cfe. Este directorio contiene cFE, es decir, la capa del sistema que ofrece diferentes servicios (Software Bus, tiempo, eventos…) que facilitan el desarrollo de módulos funcionales.

docs. Documentación oficial del proyecto CFS.

osal. Capa que permite la abstracción independientemente del OS o RTOS sobre el que se ejecute.

psp. Paquete que corrige errores de las primeras versiones y añade nueva funcionalidad al sistema (soporte para RTEMS, rendimiento mejorado de CMake, soporte para procesadores de 64-bit…).

tools. Incluye diferentes herramientas (generador de tablas, CRC, test unitarios…) que pueden ser útiles para probar el funcionamiento de la misión.

Desarrollo de una aplicación/librería

Toda la funcionalidad que se desarrolle, se incluirá en el directorio "./apps". Para identificar todos los módulos del sistema desarrollados para la misión, cada directorio se identificará utilizando la siguiente regla de nombrado:

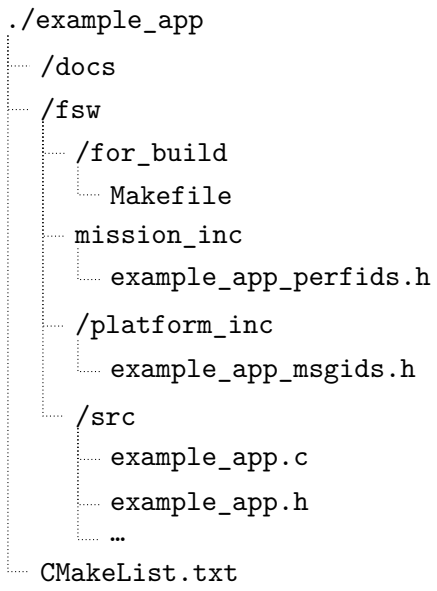


Figura C.2: Jerarquía de directorios de un módulo del sistema

nombremodulo_app | nombrelibreria_lib

A continuación, se detalla el proceso que se debe seguir para desarrollar un módulo del sistema. Todos los módulos deben de seguir la misma estructura de directorios, véase la Figura C.2.

Los ficheros y directorios que deben formar cada módulo del sistema son los siguientes:

docs. Incluye la documentación propia del módulo.

fsw. Debe incluir el código desarrollado por el usuario. Se divide en los siguientes subdirectorios:

- **for_build.** Incluye el fichero Makefile que contiene las reglas necesarias para compilar el proyecto.
- **mission_inc.** Incluye el fichero de cabecera necesario en el que se indica el identificador unívoco del módulo.
- **platform_inc.** Incluye el fichero de cabecera que contiene los identificadores unívocos de aquellos mensajes que se intercambien utilizando el Software Bus.
- **src.** Contiene el fichero con el código del sistema, además, puede incluir otra serie de ficheros de cabecera para la gestión de eventos, definición de estructuras, prototipos de las funciones…

CMakeList.txt. Este fichero incluye las reglas necesarias para la integración del módulo en el sistema cFE.

Configuración del sistema

Antes de compilar el sistema, se deben indicar aquellas aplicaciones que se incluirán en el binario que simula la CPU. Para ello, se debe modificar el fichero Makefile del directorio principal de la CPU, véase el Listado C.3.

```
# List of apps/libs to include in the build
THE_APPS :=  telemetry_app thermiccontrol_app
export THE_APPS

# List of apps that include tables for the build
THE_TBLS := scheduler
export THE_TBLS
```

Listado C.3: Fichero Makefile de la CPU

El siguiente paso que se debe realizar es indicar al simulador las diferentes opciones de las aplicaciones o librerías que se van a ejecutar. Para ello, se debe modificar el fichero "cfe_es_startup.scr" del directorio de la CPU, véase el Listado C.4.

```
CFE_APP, /cf/apps/thermiccontrol_app.so,  Thermic_AppMain,  thermiccontrol_app,  60,   8192, 0x0, 0;
CFE_APP, /cf/apps/telemetry_app.so,  Telemetry_AppMain,  TELEMETRY_APP,  90,  8192, 0x0, 0;


! Campos del script:
! 1. Tipo de objeto      -- CFE_APP para aplicaciones | CFE_LIB para librerias.
! 2. Ruta/Nombre del fichero    -- Ruta donde se encuentra el .so de la aplicacion/libreria
! 3. Punto de entrada      -- Funcion main de la aplicacion.
! 4. Nombre cFE         -- Nombre cFE de la aplicacion/libreria
! 5. Prioridad         -- Prioridad de ejecucion (solo para aplicaciones)
! 6. Tamanno de pila    -- Tamanno de pila (solo para aplicaciones)
! 7. Cargar direccion de memoria     -- Carga de direccion para aplicacion/libreria, actualmente sin implementar. Por defecto 0x0.
! 8. Accion en caso de excepcion -- Accion que debe ejecutar el sistema si ocurre alguna excepcion.
!                       0        = Reiniciar la aplicacion.
!                       Non-Zero = Realizar un reset del sistema.
!
! El sistema no leera aquello que siga al caracter '!'.
```

Listado C.4: Configuración de las aplicaciones/librerías

Compilación y ejecución del sistema

Cada vez que se desarrolle un módulo del sistema se debe compilar antes de proceder a su ejecución. La compilación y ejecución de aplicaciones que utilicen cFE requiere que algunas variables de entorno estén definidas. Para ello, se debe ejecutar el siguiente comando:

```
$ source setvars.sh
```

Listado C.5: Asignación variables de entorno

Para automatizar la tarea de compilación se ha desarrollado el siguiente script:

```
#!/bin/bash
# Se accede al directorio correspondiente
cd /build/cpu1
# Se limpia la instalacion anterior, se configuran las nuevas dependencias y se
    compila
make clean
make config
make
```

Listado C.6: Script de compilación

De la misma manera, se ha desarrollado otro script para automatizar la tarea de ejecución.

```
#!/bin/bash
# Se accede al directorio donde se encuentra el binario
cd /build/cpu1/exe
# Se ejecuta el binario con permisos de superusuario y se indica la prioridad
sudo ./core-linux.bin PO X
```

Listado C.7: Script de ejecución

La ejecución del sistema siempre se debe ejecutar con permisos de superusuario. Además, se puede seleccionar la prioridad del módulo en caso de ejecutar N módulos en paralelo, para ello se debe modificar X por la prioridad deseada, siendo 1 la prioridad más alta y N la más baja.

Si se fuerza el cierre del sistema cFE ($Ctrl + C$), la siguiente ejecución se debe ejecutar con un *flag* diferente que se encarga de corregir el comportamiento del sistema.

```
$ sudo ./core-linux.bin  --reset PO
```

Listado C.8: Ejecución tras cierre forzado

Bibliografía

[1] W. Commons, "File:sputnik asm.jpg." https://commons.wikimedia.org/wiki/File:Sputnik_asm.jpg. Accedido el 22-04-2019.

[2] W. Commons, "File:aldrin apollo 11 original.jpg." https://commons.wikimedia.org/wiki/File:Aldrin_Apollo_11_original.jpg. Accedido el 22-04-2019.

[3] W. Commons, "File:nasa mars rover.jpg." https://commons.wikimedia.org/wiki/File:NASA_Mars_Rover.jpg. Accedido el 22-04-2019.

[4] K. PEKER, "Sdlc : V model." http://koraypeker.com/2018/06/18/sdlc-v-model/. Accedido el 03-05-2019.

[5] D. Zurdo, "Los inventos de leonardo da vinci," *Manual formativo de ACTA*, no. 36, pp. 67–80, 2005.

[6] M. Paris, *From the Wright Brothers to top gun: Aviation, nationalism, and popular cinema*. Manchester University Press, 1995.

[7] I. S. in Space, "About cubesats | innovative solutions in space." https://www.isispace.nl/cubesats/. Accedido el 19-04-2019.

[8] P. Machamer, "Galileo galilei," 2005.

[9] F. J. Rutherford, "Sputnik and science education," in *Symposium "Reflecting on Sputnik: Linking the Past, Present, and Future of Educational Reform". Washington, DC*, 1997.

[10] E. S. Agency, "Historia de la exploración de marte." http://www.esa.int/esl/ESA_in_your_country/Spain/Historia_de_la_exploracion_de_Marte. Accedido el 22-04-2019.

[11] NASA, "Mars news." https://mars.nasa.gov/mer/news/. Accedido el 22-04-2019.

[12] Sputnix, "Siriussat-1." https://sputnix.ru/en/satellites-en/siriussat-1-en. Accedido el 25-04-2019.

[13] E. S. Agency, "Cubesat teams." http://www.esa.int/Education/CubeSats_-_Fly_Your_Satellite/CubeSat_teams. Accedido el 25-04-2019.

[14] E. S. Agency, "Vega." https://www.esa.int/Our_Activities/Space_Transportation/Launch_vehicles/Vega. Accedido el 25-04-2019.

[15] UPSat, "Upsat – open source – greek cubesat." https://upsat.gr/. Accedido el 25-04-2019.

[16] E. Kulu, "Nanosatellite & cubesat database." https://www.nanosats.eu. Accedido el 27-04-2019.

[17] esc Aerospace, "Spacecraft on-board computer." Accedido el 29-04-2019.

[18] E. S. Agency, "Simulus." https://www.esa.int/Our_Activities/Operations/gse/SIMULUS. Accedido el 06-11-2018.

[19] E. S. Agency, "Sentinel-3 toolbox." https://sentinel.esa.int/web/sentinel/toolboxes/sentinel-3. Accedido el 09-11-2018.

[20] E. S. Agency, "El programa copérnico." https://www.esa.int/esl/ESA_in_your_country/Spain/El_programa_Copernico. Accedido el 01-05-2019.

[21] E. S. Agency, "Sentinel-3 toolbox features | step." http://step.esa.int/main/toolboxes/sentinel-3-toolbox/s3tbx-features/. Accedido el 09-11-2018.

[22] Kubos, "Kubos: Space grade flight software and mission operations software." https://www.kubos.com/kubos/. Accedido el 13-11-2018.

[23] Kubos, "Major tom | kubos." https://www.kubos.com/majortom/. Accedido el 13-11-2018.

[24] Cubesatlab, "Cubedos." http://cubesatlab.org/CubedOS.jsp. Accedido el 14-11-2018.

[25] NASA, "Nasa - gsfc open source software." https://opensource.gsfc.nasa.gov/projects/cfe/index.php. Accedido el 26-10-2018.

[26] NASA, "Nasa operational simulation for small satellites." https://www.nasa.gov/centers/ivv/jstar/nos3.html. Accedido el 26-10-2018.

[27] R. M. R. O. UPM, "Tipos de órbitas. constelaciones de satélites." http://www.gr.ssr.upm.es/docencia/grado/csat/material/CSAT09-2-OrbitasConstelaciones.pdf. Accedido el 03-05-2019.

[28] "Telemetry and telecommand packet utilization." https://ecss.nl/standard/ecss-e-st-70-41c-space-engineering-telemetry-and-telecommand-packet-utilization-15-apr Apr 2016. Accedido el 15-10-2018.

[29] "Metodología de desarrollo de sotware. el modelo en v o de cuatro niveles.." http://www.iiia.csic.es/udt/es/blog/jrodriguez/2008/metodologia-desarrollo-sotware-modelo-en-v-o-cuatro-niveles. Accedido el 03-05-2019.

[30] B. Board, "Beagleboard-xm." https://beagleboard.org/beagleboard-xm. Accedido el 05-05-2019.

[31] Celestia.es, "Celestia." https://celestia.es/. Accedido el 29-10-2018.

[32] Tarifasgasluz, "¿conoces cuál es el precio del kwh de iberdrola 2019?." https://tarifasgasluz.com/comercializadoras/iberdrola/precio-kwh. Accedido el 14-05-2019.

[33] Vodafone, "Ofertas vodafone tv - la mejor televisión." https://www.vodafone.es/c/particulares/es/productos-y-servicios/internet-y-fijo/. Accedido el 14-05-2019.

[34] ONU, "Tratado sobre el espacio ultraterrestre." http://www.unoosa.org/pdf/publications/STSPACE11S.pdf. Accedido el 24-05-2019.

[35] ESA, "Tratados y principios sobre el espacio ultraterrestre." https://www.esa.int/About_Us/Space_Law_virtual_network_with_Latin_American_countries/Tratados_y_Principios_sobre_el_espacio_ultraterrestre. Accedido el 24-05-2019.

[36] CCSDS, "The consultative committee for space data systems (ccsds)." https://public.ccsds.org/default.aspx. Accedido el 24-05-2019.

[37] E. C. for Space Standardization. https://ecss.nl/, Nov 2014. Accedido el 24-05-2019.

[38] M. de Economía y Empresa, "Cuadro nacional de atribución de frecuencias." https://avancedigital.gob.es/espectro/Paginas/cnaf.aspx. Accedido el 25-05-2019.

[39] jalopezg uc3m, "Srs-latex-uc3m." https://github.com/jalopezg-uc3m/SRS-latex-uc3m, May 2019. Accedido el 30-05-2019.

[40] NASA, "nasa/cfe." https://github.com/nasa/cFE. Accedido el 03-10-2018.